

Intel(R) Fortran Compiler Options

Document Number: 307780-003US

Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® 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. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications. Intel may make changes to specifications and product descriptions at any time, without notice.

The software described in this document may contain software defects which may cause the product to deviate from published specifications. Current characterized software defects are available on request.

This document as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

Developers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Improper use of reserved or undefined features or instructions may cause unpredictable behavior or failure in developer's software code when running on an Intel processor. Intel reserves these features or instructions for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from their unauthorized use.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino logo, Chips, Core Inside, Dialogic, EtherExpress, ETOX, FlashFile, i386, i486, i960, iCOMP, InstantIP, Intel, Intel logo, Intel386, Intel486, Intel740, IntelDX2, IntelDX4, IntelSX2, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel XScale, IPLink, Itanium, Itanium Inside, MCS, MMX, MMX logo, Optimizer logo, OverDrive, Paragon, PDCharm, Pentium, Pentium II Xeon, Pentium III Xeon,

Intel(R) Fortran Compiler Options

Performance at Your Command, Pentium Inside, skooool, Sound Mark, The Computer Inside., The Journey Inside, VTune, Xeon, Xeon Inside and Xircom are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* Other names and brands may be claimed as the property of others.

Copyright (C) 1996-2006, Intel Corporation.

Portions Copyright (C) 2001, Hewlett-Packard Development Company, L.P.

Table Of Contents

Introduction to Compiler Options	1
How to Use This Document.....	2
New Options	4
Alphabetical Compiler Options	8
Overview of Compiler Options	8
1	11
4l2, 4l4, 4l8.....	12
4L72, 4L80, 4L132	13
4Na, 4Ya	14
4Naltparam, 4Yaltparam	15
4Nb, 4Yb	16
4Nd, 4Yd	17
4Nf	18
4Ns, 4Ys	19
4R8, 4R16.....	20
4Yf.....	21
4Yportlib.....	22
66	23
72, 80, 132.....	24
align	25
allow.....	28
altparam	30
ansi-alias, Qansi-alias.....	32

arch.....	34
architecture	36
asmattr	37
asmfile.....	39
assume	40
auto	45
auto-scalar, Qauto-scalar.....	46
autodouble	48
automatic	49
ax, Qax.....	51
B.....	53
Bdynamic	55
bintext	56
Bstatic	57
c	58
C	59
CB.....	60
ccdefault.....	61
check.....	63
cm	66
common-args	67
compile-only.....	68
complex-limited-range, Qcomplex-limited-range	69
convert	70

cpp	73
cxxlib	74
D	76
d_lines, Qd_lines	78
dbglibs.....	79
DD.....	81
debug (Linux*).....	82
debug (Windows*).....	84
debug-parameters.....	87
define	88
dll	90
double_size.....	91
dps	93
dryrun.....	94
dynamic-linker.....	95
dynamiclib	96
dyncom, Qdyncom	97
E.....	99
e90, e95	100
EP	101
error_limit.....	102
exe	103
extend_source	105
extfor	107

extfpp	108
extlnk.....	109
F (Linux*)	110
F (Windows*)	111
f66	112
f77rtl.....	113
Fa.....	115
FA	116
falias.....	117
fast	118
fcode-asm	120
Fe.....	121
fexceptions.....	122
ffnalias.....	123
Fl.....	124
finline-functions	125
finline-limit.....	127
fixed	128
fltconsistency	129
Fm.....	132
fmath-errno	133
fminshared	134
fno-omit-frame-pointer	135
fnsplit, Qfnsplit	136

Fo.....	138
fp-model, fp	139
fp-port, Qfp-port	144
fp, Oy	146
fp (Windows*).....	148
fpconstant	149
fpe	151
fpic	153
fpp, Qfpp	154
fpscomp	155
fpstkchk, Qfpstkchk.....	163
FR	164
fr32.....	165
free	166
fsource-asm	167
fsyntax-only.....	168
ftrapuv, Qtrapuv	169
ftz, Qftz.....	170
funroll-loops	172
fverbose-asm	173
fvisibility.....	174
g, Zi, Z7	177
G1, G2, G2-p9000	179
G5, G6, G7.....	180

Ge	181
gen-interfaces	182
global-hoist, Qglobal-hoist	183
Gm	184
Gs	185
Gz	186
help	187
I	188
i-dynamic.....	189
i-static.....	190
i2, i4, i8.....	191
iface	192
implicitnone	195
include.....	196
inline.....	198
inline-debug-info, Qinline-debug-info	200
inline-factor, Qinline-factor	201
inline-forceinline, Qinline-forceinline	203
inline-max-per-compile, Qinline-max-per-compile	205
inline-max-per-routine, Qinline-max-per-routine	207
inline-max-size, Qinline-max-size	209
inline-max-total-size, Qinline-max-total-size	211
inline-min-size, Qinline-min-size	213
intconstant.....	215

integer_size.....	217
ip, Qip.....	219
ip-no-inlining, Qip-no-inlining	220
ip-no-pinlining, Qip-no-pinlining	221
IPF-flt-eval-method0, QIPF-flt-eval-method0	222
IPF-fltacc, QIPF-fltacc.....	223
IPF-fma, QIPF-fma.....	224
IPF-fp-relaxed, QIPF-fp-relaxed.....	226
IPF-fp-speculation, QIPF-fp-speculation	227
ipo, Qipo.....	229
ipo-c, Qipo-c.....	231
ipo-S, Qipo-S	232
ipo-separate, Qipo-separate	233
isystem.....	234
ivdep-parallel, Qivdep-parallel	235
Kpic.....	236
I	237
L.....	238
LD	239
libdir	240
libs.....	242
link.....	245
logo	246
lowercase.....	247

map	248
map-opts, Qmap-opts	249
mcmmodel	251
MD	253
MDs.....	254
MG	255
mieee-fp	256
mixed_str_len_arg	257
ML	258
module	259
mp	260
mp1, Qprec	261
mrelax	263
MT	264
mtune	265
MW.....	267
MWs.....	268
names	269
nbs	271
no-cpprt.....	272
nobss-init, Qnobss-init	273
nodefaultlibs.....	274
nodefine	275
nofor_main	276

nostartfiles.....	277
nostdinc.....	278
nostdlib.....	279
nus	280
o	281
O	282
Ob	286
object	288
Od	289
Og	290
onetrip, Qonetrip	291
Op	292
openmp, Qopenmp	293
openmp-profile, Qopenmp-profile	295
openmp-report, Qopenmp-report	296
openmp-stubs, Qopenmp-stubs.....	298
opt-mem-bandwidth, Qopt-mem-bandwidth.....	299
opt-report, Qopt-report	301
opt-report-file, Qopt-report-file	302
opt-report-help, Qopt-report-help	303
opt-report-level, Qopt-report-level.....	304
opt-report-phase, Qopt-report-phase	305
opt-report-routine, Qopt-report-routine.....	307
optimize.....	308

Os	309
Ot	310
Ox	311
Oy	312
p	313
P	314
pad, Qpad	315
pad-source, Qpad-source	316
par-report, Qpar-report.....	318
par-threshold, Qpar-threshold.....	320
parallel, Qparallel	322
pc, Qpc.....	323
pdbfile	325
pg	326
prec-div, Qprec-div.....	327
prec-sqrt, Qprec-sqrt.....	329
prefetch, Qprefetch	330
preprocess_only.....	332
print-multi-lib	333
prof-dir, Qprof-dir	334
prof-file, Qprof-file	335
prof-format-32, Qprof-format-32.....	336
prof-gen, Qprof-gen	337
prof-gen-sampling, Qprof-gen-sampling	338

prof-genx, Qprof-genx.....	340
prof-use, Qprof-use.....	342
Qansi-alias	343
Qauto	344
Qauto-scalar	345
Qautodouble	346
Qax	347
Qchkstk	348
Qcommon-args	349
Qcomplex-limited-range.....	350
Qcpp	351
Qd_lines.....	352
Qdps	353
Qdyncom.....	354
Qextend_source.....	355
Qfp-port.....	356
Qfnsplit.....	357
Qfpp	358
Qfpstkchk.....	359
Qftz	360
Qglobal-hoist.....	361
QIA64-fr32	362
Qlfist.....	363
Qinline-debug-info.....	364

Qinline-factor.....	365
Qinline-forceinline	366
Qinline-max-per-compile.....	367
Qinline-max-per-routine	368
Qinline-max-size	369
Qinline-max-total-size	370
Qinline-min-size	371
Qinstall	372
Qip	373
Qip-no-inlining.....	374
Qip-no-pinlining.....	375
QIPF-flt-eval-method0.....	376
QIPF-fltacc.....	377
QIPF-fma	378
QIPF-fp-relaxed	379
QIPF-fp-speculation	380
Qipo	381
Qipo-c	382
Qipo-S.....	383
Qipo-separate	384
Qivdep-parallel.....	385
Qlocation.....	386
Qlowercase	388
Qmap-opts	389

Qnobss-init.....	390
Qonetrip	391
Qopenmp	392
Qopenmp-profile	393
Qopenmp-report.....	394
Qopenmp-stubs	395
Qopt-mem-bandwidth	396
Qopt-report.....	397
Qopt-report-file.....	398
Qopt-report-help.....	399
Qopt-report-level	400
Qopt-report-phase.....	401
Qopt-report-routine	402
Qoption	403
qp	405
Qpad	406
Qpad-source	407
Qpar-report	408
Qpar-threshold	409
Qparallel.....	410
Qpc	411
Qprec	412
Qprec-div	413
Qprec-sqrt.....	414

Qprefetch	415
Qprof-dir	416
Qprof-file	417
Qprof-format-32	418
Qprof-gen	419
Qprof-gen-sampling	420
Qprof-genx	421
Qprof-use	422
Qrcd	423
Qrct	424
Qsafe-cray-ptr	425
Qsave	426
Qscalar-rep	427
Qsalign	428
Qsox	429
Qssp	430
Qtcheck	431
Qtrapuv	432
Qunroll	433
Quppercase	434
Quse-asm	435
Quse_vcdebug	436
Qvc	437
Qvec-report	438

Qx	439
Qzero	440
r8, r16.....	441
rcd, Qrcd	442
real_size.....	443
recursive	445
reentrancy	446
RTCu.....	448
S.....	449
safe-cray-ptr, Qsafe-cray-ptr.....	450
save, Qsave	452
scalar-rep, Qscalar-rep	453
shared.....	454
shared-libcxa.....	455
source	457
sox, Qsox	458
ssp, Qssp	459
stand	461
static.....	463
static-libcxa	464
std, std90, std 95.....	465
syntax.....	466
syntax-only.....	467
T.....	468

tcheck, Qtcheck	469
Tf.....	470
threads	471
tpp1, tpp2, G1, G2, G2-p9000	473
tpp5, tpp6, tpp7, G5, G6, G7.....	475
traceback	477
tune	479
u (Linux* and Mac OS*)	480
u (Windows*).....	481
U	482
undefine	483
unroll, Qunroll.....	484
uppercase	485
us	486
use-asm, Quse-asm.....	487
v	488
V (Linux* and Mac OS*).....	489
V (Windows*)	490
vec-report, Qvec-report.....	491
vms	493
w	495
W0, W1	496
Wa.....	497
warn	498

watch.....	502
WB	504
what	505
winapp.....	506
WI.....	507
Wp.....	508
x, Qx.....	509
X.....	512
Xlinker	513
y	514
Z7.....	515
Zd.....	516
zero, Qzero	517
Zi	518
Zl.....	519
Zp.....	520
Zs.....	521
Cross Reference of Compiler Options.....	522
Deprecated and Removed Compiler Options.....	572
Deprecated Options	572
Removed Options	572
Related Options.....	574
Cluster OpenMP* Options (Linux only)	574
Index.....	575

Introduction to Compiler Options

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

It provides the following information:

- **New options**
This topic lists new Windows options and new Linux options in this release.
- **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 options and Linux and Mac OS 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.
- **Deprecated and removed compiler options**
This topic lists deprecated and removed options for this release. Some deprecated options show suggested replacement options.
- **Related Options**
This topic lists related options that can be used under certain conditions.

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* 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 This term refers to information that is valid on Intel®-based systems running Mac OS*.

/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 systems. For example:
Windows option: /fast
Linux and Mac OS option: -fast



Note

If an option is available on Windows systems and on Linux and Mac OS 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 option: -arch SSE

/option: keyword or
-option keyword Indicates that an option requires one of the *keyword* values.

<code>/option[: keyword]</code> or <code>-option [keyword]</code>	Indicates that the option can be used alone or with an optional keyword.
<code>option[n]</code>	Indicates that the option can be used alone or with an optional value; for example, in <code>/unroll[n]</code> or <code>-unroll[n]</code> , the <i>n</i> can be omitted or a valid value can be specified.
<code>option[-]</code>	Indicates that a trailing hyphen disables the option; for example, <code>/Qansi_alias-</code> disables the Windows option <code>/Qansi_alias</code> .
<code>[no]option</code> or <code>[no-]option</code>	Indicates that "no" or "no-" preceding an option disables the option. For example: In the Windows option <code>/[no]traceback</code> , <code>/traceback</code> enables the option, while <code>/notraceback</code> disables it. In the Linux and Mac OS option <code>-[no-]ansi-alias</code> , <code>-ansi-alias</code> enables the option, while <code>-no-ansi-alias</code> disables it. In some options, the "no" appears later in the option name; for example, <code>-fno-alias</code> disables the <code>-falias</code> 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 IA-32-based systems.
i32em	The option is available on IA-32-based systems with Intel® Extended Memory 64 Technology (Intel® EM64T).
i64	The option is available on Itanium®-based systems.

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 three 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 systems. If an option is only available on one of these operating systems, it is labeled.

Windows* Options	Description	Default
<code>/allow:[no]fpp_comments</code>	Determines how the fpp preprocessor treats Fortran end-of-line comments in preprocessor directive lines.	<code>/allow:fpp_comments</code>
<code>/assume:[no]writeable-strings</code>	Determines whether character constants go into non-read-only memory.	<code>/assume:nowriteable-strings</code>
<code>/fp:keyword</code>	Controls the semantics of floating-point calculations.	<code>/fp:fast</code>
<code>/G2-p9000</code>	Optimizes for Dual-Core	

(i64 only)	Intel® Itanium® 2 Processor 9000 Sequence processors.	OFF	
/QaxT (i32, i32em)	Optimizes for Intel® Core™2 Duo processors, Intel® Core™2 Extreme processors, and the Dual-Core Intel® Xeon® processor 5100 series.	OFF	
/Qinline-factor=<n>	Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.	OFF	
/Qinline-forceinline	Specifies that an inline routine should be inlined whenever the compiler can do so.	OFF	
/Qinline-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>	Specifies the maximum number of times the inliner may inline into a particular routine.	OFF	
/Qinline-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>	Specifies how much larger a routine can normally grow when inline expansion is performed.	OFF	
/Qinline-min-size=<n>	Specifies the upper limit for the size of what the inliner considers to be a small routine.	OFF	
/Qopt-mem-bandwidth<n> (i64 only)	Enables performance tuning and heuristics that control memory bandwidth use among processors.	/Qopt-mem-bandwidth0 for serial compilation; /Qopt-mem-bandwidth1 for parallel compilation	
/Qvc8	Specifies compatibility with Microsoft* Visual Studio 2005.	OFF	

Intel(R) Fortran Compiler Options

<code>/QxT</code> (i32, i32em)	Optimizes for Intel® Core™2 Duo processors, Intel® Core™2 Extreme processors, and the Dual-Core Intel® Xeon® processor 5100 series.	OFF
<code>/watch: [no] cmd</code>	Tells the compiler to display and execute driver tool commands.	<code>/watch:nocmd</code>
<code>/watch: [no] source</code>	Tells the compiler to display the name of the file being compiled.	<code>/watch:nosource</code>

Linux* and Mac OS* Options	Description	Default
<code>-allow</code> <code>[no] fpp_comments</code>	Determines how the fpp preprocessor treats Fortran end-of-line comments in preprocessor directive lines.	<code>-allow fpp_comments</code>
<code>-assume</code> <code>[no] writeable-strings</code>	Determines whether character constants go into non-read-only memory.	<code>-assume nowriteable-strings</code>
<code>-axT</code> (i32, i32em)	Optimizes for Intel® Core™2 Duo processors, Intel® Core™2 Extreme processors, and the Dual-Core Intel® Xeon® processor 5100 series.	OFF
<code>-check [no]uninit</code>	Determines whether checking occurs for uninitialized variables.	OFF
<code>-dynamiclib</code> (i32 only; Mac OS only)	Invokes the <code>libtool</code> command to generate dynamic libraries.	OFF
<code>-f[no-]inline-functions</code>	Enables or disables interprocedural optimizations for single file compilation.	ON
<code>-finline-limit=<i>n</i></code>	Lets you specify the maximum size of a function to be inlined.	OFF
<code>-fp-model <i>keyword</i></code>	Controls the semantics of floating-point calculations.	<code>-fp-model fast</code>
<code>-inline-factor=<<i>n</i>></code>	Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.	OFF

<code>-inline-forceinline</code>	Specifies that an inline routine should be inlined whenever the compiler can do so.	OFF
<code>-inline-max-per-compile=<n></code>	Specifies the maximum number of times inlining may be applied to an entire compilation unit.	OFF
<code>-inline-max-per-routine=<n></code>	Specifies the maximum number of times the inliner may inline into a particular routine.	OFF
<code>-inline-max-size=<n></code>	Specifies the lower limit for the size of what the inliner considers to be a large routine.	OFF
<code>-inline-max-total-size=<n></code>	Specifies how much larger a routine can normally grow when inline expansion is performed.	OFF
<code>-inline-min-size=<n></code>	Specifies the upper limit for the size of what the inliner considers to be a small routine.	OFF
<code>-isystem<dir></code>	Specifies a directory to add to the start of the system include path.	OFF
<code>-mcmmodel=<mem_model></code> (i32em only; Linux only)	Tells the compiler to use a specific memory model to generate code and store data.	<code>-mcmmodel=small</code>
<code>-mtune itanium2-p9000</code> (i64 only; Linux only)	Optimizes for Dual-Core Intel® Itanium® 2 Processor 9000 Sequence processors.	OFF
<code>-opt-mem-bandwidth<n></code> (i64 only; Linux only)	Enables performance tuning and heuristics that control memory bandwidth use among processors.	<code>/opt-mem-bandwidth0</code> for serial compilation; <code>/opt-mem-bandwidth1</code> for parallel compilation
<code>-watch [keyword]</code>	Tells the compiler to display certain information to the console output window.	<code>-nowatch</code>
<code>-xT</code> (i32, i32em)	Optimizes for Intel® Core™2 Duo processors, Intel® Core™2 Extreme processors, and the Dual-Core Intel® Xeon® processor 5100 series.	OFF

Alphabetical Compiler Options

Overview of Compiler Options

This section describes all the Linux*, Mac OS*, and Windows* compiler options.

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 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 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
- IA-32-based systems with Intel® Extended Memory 64 Technology (Intel® EM64T)
- Intel® Itanium® architecture

- Syntax

This shows the syntax on Linux and Mac OS 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".

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 Information on Compiler Options

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

- On Linux and Mac OS 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.

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 systems, `-dynamic-linkermylink` (file name) or `-umacro3` (string)
- On Windows systems, `/Famyfile.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).



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.



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.

You can print a list of the compiler options by specifying the `help` option.

1

See `one trip`, `Qonetrip`.

4l2, 4l4, 4l8

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 [warn.](#)

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: /4Yportlib

Arguments

None

Default

ON 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:recnbyte)

Data > Common Element Alignment (/align: [no] commons or
/align: [no] dcommons)

Data > SEQUENCE Types Obey Alignment Rules (/align: [no] sequence)

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -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.
recnbyte	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.

nodcommons 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 nodcommons 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 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 recnbyte	Aligns components of derived types and fields within record structures on the smaller of the size boundary specified (<i>n</i>) or the boundary that will naturally align them. <i>n</i> 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 <i>n</i> -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 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: **-noalign**
Windows: **/noalign**

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

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

align recnbyte Linux and Mac OS: **-Zp{1|2|4|8|16}**
Windows: **/Zp{1|2|4|8|16}**

align all Linux and Mac OS: **-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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
<code>allow</code> <code>nofpp_comments</code>	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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -altparam
 -noaltparam

Windows: /altparam
 /noaltparam

Arguments

None

Default

ON 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: -dps
 Windows: /Qdps, /4Yaltparam

noaltparam Linux and Mac OS: -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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-ansi-alias`
`-no-ansi-alias`

Windows: `/Qansi-alias`
`/Qansi-alias-`

Arguments

None

Default

ON 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) or `/Qansi-alias-` (Windows) so the compiler does not generate incorrect code.

Alternate Options

Linux: `-ansi_alias`

Mac OS: None

Windows: `/Qansi_alias`

arch

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T

Syntax

Linux and Mac OS: `-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 Intel® EM64T systems, only *keywords* `pn1`, `pn2`, `pn3`, and `pn4` are valid.

Alternate Options

Linux and Mac OS : None

Windows: /arch

architecture

See [arch](#).

asmattr

Specifies the contents of an assembly listing file.

IDE Equivalent

Windows: **Output > Assembler Output**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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

OFF 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
<i>/asmattr:none</i>	Produces no assembly listing. This is the same as specifying <i>/noasmattr</i> .
<i>/asmattr:machine</i>	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: None
Windows: `/noasmattr`
- `/asmattr:machine` Linux and Mac OS: `-fcode-asm`
Windows: `/FAc`
- `/asmattr:source` Linux and Mac OS: `-fsource-asm`
Windows: `/FAs`
- `/asmattr:all` Linux and Mac OS: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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

OFF 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: -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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-assume keyword`

Windows: `/assume:keyword`

Arguments

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

<code>none</code>	Disables all assume options.
<code>[no]bscc</code>	Determines whether the backslash character is treated as a C-style control character syntax in character literals.
<code>[no]buffered_io</code>	Determines whether data is immediately written to disk or accumulated in a buffer.
<code>[no]byterecl</code>	Determines whether units for the OPEN statement RECL specifier (record length) value in unformatted files are in bytes or longwords (four-byte units).
<code>[no]cc_omp</code>	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]protect_constants	Determines whether a constant actual argument or a copy of it is passed to a called routine.
[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]underscore	Determines whether the compiler appends an underscore character to external user-defined names.
[no]2underscores (Linux only)	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 <code>assume buffered_io</code> .
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 <code>-openmp</code> (Linux) or <code>/Qopenmp</code> (Windows) is specified. If compiler option <code>-openmp</code> (Linux and Mac OS) or <code>/Qopenmp</code> (Windows) is specified, the default is <code>assume cc_omp</code> .
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.
<code>protect_constants</code>	A constant actual argument is passed to a called routine. Any attempt to modify it results in an error.
<code>source_include</code>	The compiler searches for USE modules and INCLUDE files in the directory where the source file is located.
Windows: <code>nunderscore</code> Linux and Mac OS: <code>underscore</code>	On Windows systems, the compiler does not append an underscore character to external user-defined names. On Linux and Mac OS systems, the compiler appends an underscore character to external user-defined names.
<code>no2underscores</code> (Linux only)	The compiler does not append two underscore characters to external user-defined names that contain an embedded underscore.
<code>nowriteable-strings</code>	The compiler puts character constants into read-only memory.

Description

This option specifies assumptions to be made by the compiler.

Option	Description
<code>assume none</code>	Disables all the assume options.
<code>assume bscc</code>	Tells the compiler to treat the backslash character (\) as a C-style control (escape) character syntax in character literals. The "bscc" keyword means "BackSlashControlCharacters."
<code>assume buffered_io</code>	<p>Tells the compiler to accumulate records in a buffer. This sets the default for opening sequential output files to <code>BUFFERED='YES'</code>, which also occurs if the <code>FORT_BUFFERED</code> run-time environment variable is specified.</p> <p>When this option is specified, the internal buffer is filled, possibly by many record output statements (<code>WRITE</code>), 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.</p> <p>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.</p> <p>The <code>OPEN</code> statement <code>BUFFERED</code> specifier applies to a specific logical unit. In contrast, the <code>assume [no]buffered_io</code> option and the <code>FORT_BUFFERED</code> environment variable apply to all Fortran units.</p>
<code>assume byterecl</code>	Specifies that the units for the <code>OPEN</code> statement <code>RECL</code> specifier (record length) value are in bytes for unformatted data files, not longwords (four-byte units). For formatted files, the <code>RECL</code> 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
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
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.

`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.

Intel(R) Fortran Compiler Options

`assume 2underscores` (Linux only) 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 nobsc` Linux and Mac OS: `-nbs`
Windows: `/nbs`

`assume dummy_aliases` Linux and Mac OS: `-common-args`
Windows: `/Qcommon-args`

`assume underscore` Linux and Mac OS: `-us`
Windows: `/us`

`assume nounderscore` Linux and Mac OS: `-nus`
Windows: None

auto

See [automatic](#).

auto-scalar, Qauto-scalar

Causes allocation of scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL to the run-time stack.

IDE Equivalent

Windows: **Data > Local Variable Storage** (/Qsave, /Qauto, /Qauto_scalar)

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -auto-scalar

Windows: /Qauto-scalar

Arguments

None

Default

ON 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), 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 or 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

Linux: `-auto_scalar`

Mac OS: None

Windows: `/Qauto_scalar`

See Also

`auto` compiler option

`save` compiler option

autodouble

See [real_size](#).

automatic

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

IDE Equivalent

Windows: **Data > Local Variable Storage**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-automatic`
`-noautomatic`

Windows: `/automatic`
`/noautomatic`

Arguments

None

Default

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

Description

This option places local variables, except those declared as SAVE, on 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 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) or `/Qsave` (Windows).



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: `-auto`
Windows: `/auto`, `/Qauto`, `/4Ya`

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

See Also

`auto-scalar` compiler option

`save`, `Qsave` compiler option

ax, Qax

Directs the compiler to generate processor-specific code if there is a performance benefit, while also generating generic IA-32 code.

IDE Equivalent

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

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T

Syntax

Linux and Mac OS: `-axprocessor`

Windows: `/Qaxprocessor`

Arguments

processor Is the processor for which you want to target your program. Possible values are:

- K** Code is optimized for Intel® Pentium® III and compatible Intel processors.
- W** Code is optimized for Intel Pentium 4 and compatible Intel processors.
- N** Code is optimized for Intel Pentium 4 and compatible Intel processors.
This option also enables new optimizations in addition to Intel processor-specific optimizations.
- B** Code is optimized for Intel Pentium M and compatible Intel processors.
This option also enables new optimizations in addition to Intel processor-specific optimizations.
- P** Code is optimized for Intel® Core™ Duo processors, Intel® Core™ Solo processors, Intel® Pentium® 4 processors with Streaming SIMD Extensions 3, and compatible Intel processors with Streaming SIMD Extensions 3. This option also enables new optimizations in addition to Intel processor-specific optimizations.
- T** Code is optimized for Intel® Core™2 Duo processors, Intel® Core™2 Extreme processors, and the Dual-Core Intel® Xeon® processor 5100 series.

Default

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

Description

This option directs the compiler to generate processor-specific code if there is a performance benefit, while also generating generic IA-32 code. The generic code is usually slower than the specialized code.

It 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. The generic version will run on any IA-32 processor.

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 IA-32 processors.

On Intel® EM64T systems, `W`, `P`, and `T` are the only valid *processor* values.

On Mac OS systems, `P` is the only valid *processor* value. On these systems, `-axP` is equivalent to `-xP`, which is the default and is always set.

You can use more than one of the *processor* values by combining them. For example, you can specify `-axNB` (Linux) or `/QaxNB` (Windows) to generate code for Intel® Pentium® 4 processors and Intel Pentium M processors.

If you specify both the `-ax` and `-x` options (Linux) 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.

Alternate Options

None

See Also

`x` compiler option

B

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-Bdir`

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-Bdynamic`

Mac OS: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

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

Arguments

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

Default

OFF 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: None

Windows: `/Vstring`

Bstatic

Enables static linking of a user's library.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-Bstatic`

Mac OS: 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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: None

Windows: `/compile-only`, `/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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-ccdefault keyword`

Windows: `/ccdefault:keyword`

Arguments

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

- `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 (/RTCu)

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)

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

Windows: /check[:keyword]
/nocheck

Arguments

keyword Specifies the run-time conditions to check. Possible values are:

none	Prevents all run-time checking.
[no] arg_temp_created	Determines whether checking occurs for actual arguments before routine calls.
[no] bounds	Determines whether checking occurs for bounds of array subscript and substring references.
[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] uninit	Determines whether checking occurs for uninitialized variables.
all	Enables all check options.

Default

OFF 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
<code>check none</code>	Prevents all checking for run-time failures. Disables all check options (same as <code>nocheck</code>).
<code>check arg_temp_created</code>	Generates code to check if actual arguments are copied into temporary storage before routine calls. If a copy is made at run-time, an informative message is displayed.
<code>check bounds</code>	Generates code to perform run-time checks on 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 <code>*</code> 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.
<code>check format</code>	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 <code>REAL*4</code> item formatted with an <code>I</code> edit descriptor). With <code>check noformat</code> , 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 <code>INTEGER*2</code> item to an <code>E</code> edit descriptor).
<code>check output_conversion</code>	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 (<code>*</code>), and execution continues.
<code>check uninit</code>	Generates code to check for uninitialized variables. If a variable is read before written, a run-time error routine will be called.
<code>check all</code>	Enables all check options. This is the same as specifying <code>check</code> with no keyword.

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

Alternate Options

<code>check none</code>	Linux and Mac OS: <code>-nocheck</code> Windows: <code>/nocheck, /4Nb</code>
<code>check bounds</code>	Linux and Mac OS: <code>-CB</code> Windows: <code>/CB</code>
<code>check uninit</code>	Linux and Mac OS: <code>None</code> Windows: <code>/RTCu</code>
<code>check all</code>	Linux and Mac OS: <code>-check, -C</code> Windows: <code>/check, /4Yb, /C</code>

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

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

Arguments

None

Default

OFF 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

Linux: `-complex_limited_range`

Mac OS: None

Windows: `/Qcomplex_limited_range`

convert

Specifies the format of unformatted files containing numeric data.

IDE Equivalent

Windows: **Compatibility > Unformatted File Conversion**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-convert keyword`

Windows: `/convert:keyword`

Arguments

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

<code>native</code>	Specifies that unformatted data should not be converted.
<code>big_endian</code>	Specifies that the format will be big endian for integer data and big endian IEEE floating-point for real and complex data.
<code>cray</code>	Specifies that the format will be big endian for integer data and CRAY* floating-point for real and complex data.
<code>fdx</code>	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.
<code>fgx</code>	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.
<code>ibm</code>	Specifies that the format will be big endian for integer data and IBM* System\370 floating-point format for real and complex data.
<code>little_endian</code>	Specifies that the format will be little endian for integer data and little endian IEEE floating-point for real and complex data.
<code>vaxd</code>	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.
<code>vaxg</code>	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

Intel(R) Fortran Compiler Options

or COMPLEX*32.

Alternate Options

None

cpp

See [fpp](#), [Qfpp](#).

cxxlib

Tells the compiler to link using certain C++ run-time libraries.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-cxxlib[-mode]`
`-no-cxxlib`

Windows: None

Arguments

mode Specifies which C++ run-time libraries to use. Possible values are:

`gcc[=dir]` Tells the compiler to link using the C++ run-time libraries provided by gcc.
dir is an optional top-level location for the gcc binaries and libraries.

`icc` Tells the compiler to link using the C++ run-time libraries provided by Intel. `-cxxlib-icc` is only available on IA-32 and Itanium®-based Linux systems.

Default

`-cxxlib-gcc` On Mac OS* systems, the compiler uses the run-time libraries provided by gcc.

`-no-cxxlib` On Linux* systems, 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 certain C++ run-time libraries.

If you specify the option with no *mode*, the compiler uses the default C++ libraries.



For full interoperability with gcc, use `-cxxlib-gcc`. Do not use the `-cxxlib-gcc` option if your version of gcc is older than 3.2.



All object binaries used in a single link must be compiled with the same *mode* value, whether invoked explicitly or by default.

Alternate Options

`-no-cxxlib` Linux: `-no-cpprt`, `-no_cpprt`
Mac OS: None
Windows: None

D

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

IDE Equivalent

Windows:

General: >Preprocessor Definitions (/define)

Preprocessor>Preprocessor Definitions (/define)

Preprocessor > Preprocessor Definitions to FPP only (/nodf)

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

Windows: `/Dname [=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, `Dname="string"`.

Default

OFF 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`.



On Linux and Mac OS 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: None
Windows: `/define`

`noD` Linux and Mac OS: `-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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-d_lines`

Windows: `/d_lines`
 `/Qd_lines`

Arguments

None

Default

OFF Debug lines are treated as comment lines.

Description

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: `-DD`

Windows: None

dbglibs

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: /dbglibs
 /nodbglibs

Arguments

None

Default

OFF 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:
Programming with Mixed Languages Overview

DD

See [d_lines](#), [Qd_lines](#).

debug (Linux*)

Specifies settings that enhance debugging.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-debug keyword`

Mac OS: None

Windows: None

Arguments

keyword is an option for enhanced debugging. Possible values are:

<code>[no]inline_debug_info</code>	Determines whether enhanced debug information is produced for inlined code.
<code>[no]semantic_stepping</code>	Determines whether enhanced debug information useful for breakpoints and stepping is produced.
<code>[no]variable_locations</code>	Determines whether enhanced debug information useful in finding scalar local variables is produced.
<code>extended</code>	Enables <code>semantic_stepping</code> and <code>variable_locations</code> .

Default

OFF No enhanced debugging information is produced.

Description

This option specifies settings that enhance debugging.

To use this option, you must also specify the `-g` option.

Option	Description
<code>-debug</code> <code>inline_debug_info</code>	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.
<code>-debug semantic_stepping</code>	<p>Produces enhanced debug information useful for breakpoints and stepping.</p> <p>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.</p>
<code>-debug variable_locations</code>	<p>Produces enhanced debug information useful in finding scalar local variables.</p> <p>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.</p>
<code>-debug extended</code>	Sets the debug options <code>semantic_stepping</code> and <code>variable_locations</code> .

Alternate Options

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

See Also

`debug` (Windows*) compiler option

Building Applications: Debugging Overview

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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:minimal 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
<code>/debug:none</code>	Produces no symbol table information. It is the same as specifying <code>/nodebug</code> . This <code>/debug</code> option produces the smallest size object module and passes <code>/debug:none</code> to the linker.
<code>/debug:minimal</code>	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 <code>/debug:none</code> , but is smaller than if you specified <code>/debug:full</code> . This option passes <code>/debug:minimal</code> to the linker.
<code>/debug:partial</code>	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 <code>/debug:none</code> , but is smaller than if you specified <code>/debug:full</code> . This option passes <code>/debug:partial</code> to the linker. Note: This option is not available in the IDE.
<code>/debug:full</code> or <code>/debug</code>	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 <code>/debug:full</code> to the linker. If you specify <code>/debug:full</code> for an application that makes calls to C library routines and you need to debug calls into the C library, you should also specify <code>/dbglibs</code> to request that the appropriate C debug library be linked against.
<code>/debug:semantic_stepping</code>	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.
<code>/debug:extended</code>	Enables the debug option <code>semantic_stepping</code> .

Alternate Options

`/debug:minimal` Linux and Mac OS: None

Intel(R) Fortran Compiler Options

Windows: `/Zd` (this is a deprecated option)

`/debug:full` or `/debug` Linux and Mac OS: None

Windows: `/Zi`, `/Z7`

See Also

`dbglibs` compiler option

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

debug-parameters

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

Specifies a definition (symbol) to use with conditional compilation.

IDE Equivalent

Windows:

General > Preprocessor Definitions

Preprocessor > Preprocessor Definitions

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: `/define:name [=value]`

Arguments

name Is the name of the definition (symbol).

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

Default

OFF No additional symbols are defined.

Description

This option specifies a definition (symbol) to use with conditional compilation directives or the Fortran preprocessor (`fpp`). You can assign an optional value to the definition.

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

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

If you want the symbol values defined to apply only to `fpp` and not to compiler directives, you must also specify option `/noD` on the command line.

Alternate Options

Linux and Mac OS: `-D`

Windows: `/D`

See Also

Building Applications: Predefined Preprocessor Symbols

dll

Specifies that a program should be linked as a dynamic-link (DLL) library.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: None

Windows: /LD

double_size

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

IDE Equivalent

Windows: **Data > Default Double-Precision KIND**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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
<code>double_size 64</code>	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).
<code>double_size 128</code>	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

Intel(R) Fortran Compiler Options

None

dps

See [altparam](#).

dryrun

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

dynamic-linker

Specifies a dynamic linker other than the default.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-dynamic-linker file`

Mac OS: 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

Syntax

Linux: None

Mac OS: `-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 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: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-dyncom "common1, common2, common3"`

Windows: `/Qdyncom "common1, common2, common3"`

Arguments

common1, common2, common3 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 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

See [warn.](#)

EP

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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: **Diagnostics > Error Limit**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: `-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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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
<code>extend_source 72</code>	Specifies that the statement field ends at column 72.
<code>extend_source 80</code>	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: -72
Windows: /4L72

`extend_source 80` Linux and Mac OS: -80
Windows: /4L80

`extend_source 132` Linux and Mac OS: -132
Windows: /Qextend_source, /4L132

extfor

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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 (Linux*)

See [preprocess_only](#).

F (Windows*)

Specifies the stack reserve amount for the program.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: `/Fn`

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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: `-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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -f77rtl
-nof77rtl

Windows: /f77rtl
/nof77rtl

Arguments

None

Default

OFF 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

fast

Maximizes speed across the entire program.

IDE Equivalent

Windows: **General > Optimization**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 Itanium®-based systems:
Windows: `/O3` and `/Qipo`
Linux: `-ipo`, `-O3`, and `-static`
- On IA-32 and Intel® EM64T systems:
Mac OS: `-ipo`, `-O3`, `-no-prec-div`, and `-static`
Windows: `/O3`, `/Qipo`, `/Qprec-div-`, and `/QxP`
Linux: `-ipo`, `-O3`, `-no-prec-div`, `-static`, and `-xP`
Note that programs compiled with the `-xP` (Linux) or `/QxP` (Windows) option will detect non-compatible processors and generate an error message during execution.

On IA-32 and Intel® EM64T systems, the `-xP` or `/QxP` 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 `-xP` or `/QxP` 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 `-xP` or `/QxP` 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

See keyword `machine` in [asmattr](#)

Fe

See [exe.](#)

fexceptions

Enables exception handling table generation.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and `-fexceptions`
Mac OS: `-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 certain interprocedural optimizations for single file compilation.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

Windows: None

Arguments

None

Default

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

Description

This option enables certain interprocedural optimizations for single file compilation. These optimizations are a subset of full intra-file interprocedural optimizations.

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

None

See Also

`ip, Qip` compiler option

`finline-limit` compiler option

Intel(R) Fortran Compiler Options

Optimizing Applications:

Compiler Directed Inline Expansion of User Functions

Criteria for Inline Function Expansion

finline-limit

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

fixed

Specifies source files are in fixed format.

IDE Equivalent

Windows: **Language > Source File Format**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 systems, file names and file extensions are case sensitive.

Alternate Options

Linux and Mac OS: `-FI`

Windows: `/nofree`, `/FI`, `/4NF`

fltconsistency

Enables improved floating-point consistency.

IDE Equivalent

Windows: **Floating-Point > Floating-Point Consistency** (/Op)

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

Windows: `/fltconsistency`
`/nofltconsistency`

Arguments

None

Default

OFF 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 IA-32 systems and Intel® EM64T systems, floating-point user variables are not assigned to registers.
- On Itanium®-based systems, 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) 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) or `/fp` (Windows).

Alternate Options

`fltconsistency` Linux and Mac OS: `-mp, -mieee-fp`
 Windows: `/Op`

`nofltconsistency` Linux and Mac OS: `-mno-ieee-fp`

Windows: None

See Also

`mp1`, `Qprec` compiler option

`fp-model`, `fp` compiler option

Optimizing Applications: Floating-point Options for Itanium(R)-based Applications

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

Windows: None

Arguments

None

Default

OFF 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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.

Since symbols defined in the main program cannot be preempted, this allows the compiler to treat symbols declared with default visibility as though they have protected visibility (so `-fminshared` implies `-fvisibility=protected`).

Also, the compiler need not 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

fno-omit-frame-pointer

See [fp](#), [Oy](#).

fnsplit, Qfnsplit

Enables function splitting.

IDE Equivalent

None

Architectures

`/Qfnsplit [-]`: IA-32, Intel® Itanium® architecture

`- [no-] fnsplit`: Itanium® architecture

Syntax

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

Mac OS: None

Windows: `/Qfnsplit`
 `/Qfnsplit-`

Arguments

None

Default

OFF 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

Fo

See [object](#).

fp-model, fp

Controls the semantics of floating-point calculations.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

Windows: `/fp:keyword`

Arguments

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

<code>precise</code>	Enables value-safe optimizations on floating-point data and rounds intermediate results to source-defined precision.
<code>fast [=1 2]</code>	Enables more aggressive optimizations on floating-point data.
<code>strict</code>	Enables <code>precise</code> and <code>except</code> , disables contractions, enables the property that allows modification of the floating-point environment.
<code>source</code>	Enables value-safe optimizations on floating-point data and rounds intermediate results to source-defined precision.
<code>[no-]except</code> (Linux and Mac OS) or <code>except [-]</code> (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 `-00` (Linux and Mac OS) or `/Od` (Windows), the default is `fltconsistency`.

Description

This option controls the semantics of floating-point calculations.

The *keywords* 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
<code>-fp-model precise</code> or <code>/fp:precise</code>	<p>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.</p> <p>The compiler assumes the default floating-point environment; you are not allowed to modify it.</p> <p>Floating-point exception semantics are disabled by default. To enable these semantics, you must also specify <code>-fp-model except</code> or <code>/fp:except</code>.</p> <p>This keyword is equivalent to keyword <i>source</i>.</p> <p>For information on the semantics used to interpret floating-point calculations in the source code, see <i>precise</i> in Examples.</p>
<code>-fp-model fast [=1 2]</code> or <code>/fp:fast [=1 2]</code>	<p>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.</p> <p>Specifying <i>fast</i> is the same as specifying <i>fast=1</i>. <i>fast=2</i> may produce faster and less accurate results.</p> <p>Floating-point exception semantics are disabled by default and they cannot be enabled because you cannot specify <i>fast</i> and <i>except</i> together in the same compilation. To enable exception semantics, you must explicitly specify another keyword (see other keyword descriptions for details).</p> <p>For information on the semantics used to interpret floating-point</p>

	calculations in the source code, see <i>fast</i> in Examples.
<code>-fp-model strict</code> or <code>/fp:strict</code>	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 <code>-fp-model no-exception</code> or <code>/fp:exception-</code> . For information on the semantics used to interpret floating-point calculations in the source code, see <i>strict</i> in Examples.
<code>-fp-model source</code> or <code>/fp:source</code>	This option is equivalent to <i>keyword</i> <code>precise</code> . 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 <i>precise</i> 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 <i>source</i> in Examples.
<code>-fp-model except</code> or <code>/fp:except</code>	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

The included examples show:

- A small example of source code
Note that the same source code is considered in all the included examples.
- The semantics that are used to interpret floating-point calculations in the source code
- One or more possible ways the compiler may interpret the source code
Note that there are several ways the compiler may interpret the code; we show just some of these possibilities.

Intel(R) Fortran Compiler Options

Examples are provided for the following keywords:

- fast
- precise (see example for source)
- source
- strict

-fp-model fast Or /fp:fast

Example source code:

```
REAL T0, T1, T2;  
...  
T0 = 4.0E + 0.1E + T1 + T2;
```

When this option is specified, the compiler applies the following semantics:

- Additions may be performed in any order
- Intermediate expressions may use single, double, or extended precision
- The constant addition may be pre-computed, assuming the default rounding mode

Using these semantics, the following shows some possible ways the compiler may interpret the original code:

```
REAL T0, T1, T2;  
...  
T0 = (T1 + T2) + 4.1E;
```

```
REAL T0, T1, T2;  
...  
T0 = (T1 + 4.1E) + T2;
```

-fp-model source Or /fp:source

This setting is equivalent to -fp-model precise or /fp:precise.

Example source code:

```
REAL T0, T1, T2;  
...  
T0 = 4.0E + 0.1E + T1 + T2;
```

When this option is specified, the compiler applies the following semantics:

- Additions are performed in program order, taking into account any parentheses
- Intermediate expressions use the precision specified in the source code

- The constant addition may be pre-computed, assuming the default rounding mode

Using these semantics, the following shows a possible way the compiler may interpret the original code:

```
REAL T0, T1, T2;
...
T0 = ((4.1E + T1) + T2);
```

-fp-model strict **or** **/fp:strict**

Example source code:

```
REAL T0, T1, T2;
...
T0 = 4.0E + 0.1E + T1 + T2;
```

When this option is specified, the compiler applies the following semantics:

- Additions are performed in program order, taking into account any parentheses
- Expression evaluation matches expression evaluation under keyword `precise`
- The constant addition will not be pre-computed, because there is no way to tell what rounding mode will be active when the program runs.

Using these semantics, the following shows a possible way the compiler may interpret the original code:

```
REAL T0, T1, T2;
...
T0 = REAL (((REAL)4.0E + (REAL)0.1E) + (REAL)T1) + (REAL)T2);
```

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.

Building Applications: Native IEEE* Floating-Point Representations Overview

Optimizing Applications: Floating-point Arithmetic Optimizations Overview

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: None

Architectures

-fp-port: IA-32, Intel® EM64T

/Qfp-port: IA-32

Syntax

Linux and Mac OS: -fp-port
 -no-fp-port

Windows: /Qfp-port
 /Qfp-port-

Arguments

None

Default

Linux and Mac OS: ON On Linux and Mac OS systems, floating-point results are rounded after floating-point operations; on Windows systems, they are not.

Windows: OFF

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.

On Windows systems, the default is to keep results of floating-point operations in higher precision; to get this behavior on Linux and Mac OS systems, specify -no-fp-port. This provides better performance but less consistent floating-point results.

Alternate Options

Linux: -fp_port

Mac OS: None

Windows: /Qfp_port

See Also

Optimizing Applications: Floating-point Options for IA-32 and Intel(R) EM64T

fp, Oy

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

IDE Equivalent

Windows: **Optimization > Omit Frame Pointers**

Linux: None

Mac OS: None

Architectures

-fp: IA-32, Intel® EM64T

/Oy[-]: IA-32

Syntax

Linux and Mac OS: -fp

Windows: /Oy
 /Oy-

Arguments

None

Default

Linux and Mac OS: OFF On Windows* systems, EBP is used as a general-purpose register in optimizations; on Linux* and Mac OS systems, it is not.

Windows: ON

Description

Option -fp disallows, while option /Oy allows, use of EBP as a general-purpose register in optimizations.

Some debuggers expect EBP to be used as a stack frame pointer, and cannot produce a stack backtrace unless this is so. The -fp and /Oy- options disallow the use of the EBP register in optimizations and 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 -fp: turning off optimizations with -O0
- For /Oy-: turning off /O1, /O2, or /O3 optimizations

The `-fp` option is set when you specify the `-O0` option. On IA-32 systems and Intel EM64T systems, it is also set when you specify the `-g` option. If you specify the `-O1`, `-O2`, or `-O3` option, `-fp` is OFF.

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

Using the `-fp` 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: `-fno-omit-frame-pointer`

Windows: None

fp (Windows*)

See [fp-model](#), [fp](#).

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

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-fpconstant`
`-nofpconstant`

Windows: `/fpconstant`
`/nofpconstant`

Arguments

None

Default

OFF 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-fpen`

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) or `/Qftz` (Windows) option; therefore underflow results will be set to zero unless you explicitly specify `-no-ftz` (Linux and Mac OS) or `/Qftz-` (Windows).

On Itanium®-based systems, underflow behavior is equivalent to specifying option `-ftz` or `/Qftz`.

On IA-32 or Intel® EM64T systems, 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. This option sets the `-ftz` or `/Qftz` option; therefore underflow results will be set to zero unless you explicitly specify `-no-ftz` or `/Qftz-`.

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, 03`, or `02` on IA-32 and Intel EM64T systems. 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

Building Applications: Using the Floating-Point Exception Handling (`-fpe`) Option
Optimizing Applications: Understanding Floating-point Performance

fpic

Tells the compiler to generate position-independent code.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

Mac OS: None

Windows: None

Arguments

None

Default

OFF 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 IA-32 systems and Intel® EM64T systems, this option must be used when building shared objects.

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

Alternate Options

Linux: `-Kpic`, `-KPIC` (these are deprecated options)

Mac OS: None

Windows: None

fpp, Qfpp

Runs the Fortran preprocessor on source files before compilation.

IDE Equivalent

Windows: **Preprocessor > Preprocess Source File**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -fpp
-nofpp

Windows: /fpp
/nofpp
/Qfppn

Arguments

n Only available for /Qfpp. Tells the compiler whether to run the preprocessor or not. Possible values are 0 (meaning do not run the compiler) or any number greater than 0 (meaning run the compiler).

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.

For Windows option /Qfpp, /Qfpp0 is equivalent to /nofpp and /Qcpp0. When *n* is a number greater than 0, /Qfppn (and /Qcppn) are equivalent to /fpp.

Alternate Options

Linux and Mac OS: -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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -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.

<code>[no]logicals</code>	Determines what compatibility is used for representation of LOGICAL values.
<code>all</code>	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
<code>fpscomp none</code>	Specifies that no options should be used for compatibility with Fortran PowerStation. This is the same as specifying <code>nofpscomp</code> . Option <code>fpscomp none</code> enables full Intel® Fortran compatibility. If you omit <code>fpscomp</code> , the default is <code>fpscomp libs</code> . You cannot use the <code>fpscomp</code> and <code>vms</code> options in the same command.
<code>fpscomp filesfromcmd</code>	Specifies Fortran PowerStation behavior when the OPEN statement FILE= specifier is blank (FILE= ' '). It causes the following actions to be taken at run-time: <ul style="list-style-type: none"> • 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.

fpscomp
ioformat

Specifying `fpscomp general sets fpscomp ldio_spacing`.

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 $1 \leq N < 10^{**7}$, use F15.6 for single precision or F24.15 for double.

For $N < 1$ or $N \geq 10^{**7}$, 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 $1 \leq N < 10^{**7}$, use F15.6 for single precision or F24.15 for double.

For $N < 1$ or $N \geq 10^{**7}$, 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.

<code>fpscomp nolib</code>	Prevents the portability library from being passed to the linker.
<code>fpscomp ldio_spacing</code>	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 <code>fpscomp general</code> , it sets <code>fpscomp ldio_spacing</code> .
<code>fpscomp logicals</code>	Specifies that integers with a non-zero value are treated as true, integers with a zero value are treated as false. The literal constant <code>.TRUE.</code> has an integer value of 1, and the literal constant <code>.FALSE.</code> 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 <code>fpscomp nologicals</code> , 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 <code>.TRUE.</code> has an integer value of -1, and the literal constant <code>.FALSE.</code> 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 <code>fpscomp logical</code> option affects the results of all logical expressions and affects the return value for the following Fortran features: <ul style="list-style-type: none"> • 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
<code>fpscomp all</code>	Specifies that all options should be used for compatibility with Fortran PowerStation. This is the same as specifying <code>fpscomp</code> with no keyword. Option <code>fpscomp all</code> enables full compatibility with Fortran PowerStation.

Alternate Options

None

See Also

Building Applications: Microsoft Fortran PowerStation Compatible Files

fpstkchk, Qfpstkchk

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: None

Architectures

IA-32, Intel® EM64T

Syntax

Linux and Mac OS: `-fpstkchk`

Windows: `/Qfpstkchk`

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

None

FR

See [free](#).

fr32

Disables the use of the high floating-point registers.

IDE Equivalent

None

Architectures

Intel® Itanium® architecture

Syntax

Linux: `-fr32`

Mac OS: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -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: -FR

Windows: /nofixed, /FR, /4Yf

See Also

fixed compiler option

fsource-asm

See keyword `source` in [asmattr](#).

fsyntax-only

See [syntax-only](#).

ftrapuv, Qtrapuv

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

IDE Equivalent

Windows: **Data > Initialize Local Variables to NaN**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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) 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: **Floating-Point > Flush Denormal Results to Zero**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-ftz`
`-no-ftz`

Windows: `/Qftz`
`/Qftz-`

Arguments

None

Default

OFF The compiler lets results gradually underflow.

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. Intel Itanium systems have FTZ but not DAZ. Intel EM64T systems have both FTZ and DAZ. FTZ and DAZ are not supported on all IA-32 architectures.

When `-ftz` (Linux and Mac OS) or `/Qftz` (Windows) is used in combination with an SSE-enabling option on IA-32 systems (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 (this processor check fails for non-Intel machines). `-no-ftz` (Linux and Mac OS) 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) or `/fpe:0` and `/fpe:1` (Windows) set `-ftz` or `/Qftz`. On Itanium®-based systems, optimization option `O3` sets this option. Optimization option `O2` sets the `-no-ftz` or `/Qftz-` option.

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

When SSE instructions are used on IA-32 systems, options `-no-ftz` and `/Qftz-` are ignored. However, you can enable gradual underflow by calling a function in C in the main program that clears the FTZ and DAZ bits in the MXCSR or by calling the function `_set_fpe` in the main program to clear those bits. Be aware that denormal processing can significantly slow down computation.

Alternate Options

None

See Also

`x`, `Qx` compiler option

Building Applications: Using the Floating-Point Exception Handling (/fpe) 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-fverbose-asm`
`-fno-verbose-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 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-fvisibility=keyword`
`-fvisibility-keyword=file`

Windows: None

Arguments

keyword Specifies the visibility setting. Possible values are:

<code>default</code>	Sets visibility to default.
<code>extern</code>	Sets visibility to extern.
<code>hidden</code>	Sets visibility to hidden.
<code>internal</code>	Sets visibility to internal.
<code>protected</code>	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`). For symbols specified in *file*, the second syntax form overrides the first.



Note

These two ways to explicitly set visibility are mutually exclusive. You may set visibility in the declaration, or specify the symbol name in a file, but not both.

Option	Description
<code>-fvisibility=default</code> <code>-fvisibility-</code> <code>default=file</code>	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.
<code>-fvisibility=extern</code> <code>-fvisibility-</code> <code>extern=file</code>	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.
<code>-fvisibility=hidden</code> <code>-fvisibility-</code> <code>hidden=file</code>	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.
<code>-fvisibility=internal</code> <code>-fvisibility-</code> <code>internal=file</code>	Sets visibility of symbols to internal. This means the symbol cannot be referenced outside its defining component, either directly or indirectly.
<code>-fvisibility=protected</code> <code>-fvisibility-</code> <code>protected=file</code>	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*)

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -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) or `o3` (Windows) the default unless `o2` (or another `o` option) is explicitly specified in the same command line.

On Intel® EM64T Linux systems and Linux and Mac OS IA-32 systems, specifying the `g` or `o0` option sets the `fp` option.

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

Alternate Options

Linux and Mac OS: None

Windows: `/debug:full` (or `/debug`)

See Also

`zd` compiler option

G1, G2, G2-p9000

See [tpp1](#), [tpp2](#), [G1](#), [G2](#), [G2-p9000](#).

G5, G6, G7

See [tpp5](#), [tpp6](#), [tpp7](#), [G5](#), [G6](#), [G7](#).

Ge

Enables stack-checking for all functions.
This option has been deprecated.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: **External Procedures > Generate Interface Blocks**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-gen-interfaces`
`-nogen-interfaces`

Windows: `/gen-interfaces`
`/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.

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-global-hoist`
`-no-global-hoist`

Windows: `/Qglobal-hoist`
`/Qglobal-hoist-`

Arguments

None

Default

ON 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) 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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](#).

help

Displays the list of compiler options.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-help`

Windows: `/help`

Arguments

None

Default

OFF No list is displayed unless this compiler option is specified.

Description

This option displays the list of available compiler options.

Alternate Options

Linux and Mac OS: None

Windows: `/?`

I

Specifies a directory to add to the include path.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-Idir`

Windows: `/Idir`

Arguments

dir Is the directory to add to the include path.

Default

OFF The default include path is used.

Description

This option specifies a directory to add to the include path, which is searched for module files referenced in USE statements and include files referenced in INCLUDE statements.

For more information on Windows systems, see `include`.

Alternate Options

Linux and Mac OS: None

Windows: `/include`

i-dynamic

Causes Intel-provided libraries to be linked in dynamically.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-i-dynamic`

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 `-i-static`.

Alternate Options

Linux: `-i_dynamic`

Mac OS: None

Windows: None

See Also

`i-static` compiler option

i-static

Causes Intel-provided libraries to be linked in statically.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-i-static`

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 `-i-dynamic`.

Alternate Options

Linux: `-i_static`

Mac OS: None

Windows: None

See Also

`i-dynamic` compiler option

i2, i4, i8

See [integer_size](#).

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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 IA-32 and Intel EMT64 systems, 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 Itanium®-based systems, the only option available is `/iface:default`.

Option	Description
<code>/iface:default</code>	Tells the compiler to use the default calling conventions. This is the only option available on Itanium®-based systems.
<code>/iface:cref</code>	Tells the compiler to use calling conventions <code>C</code> , <code>REFERENCE</code> .
<code>/iface:cvf</code>	Tells the compiler to use calling convention <code>CVF</code> (Compaq* and Powerstation* compatibility). By default, <code>/iface:cvf</code> passes arguments by reference. <code>/iface:cvf</code> sets the <code>/iface:mixed_str_len_arg</code> option. This causes <code>CHARACTER</code> variables to be passed as address/length pairs.
<code>/iface:mixed_str_len_arg</code>	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.
<code>/iface:stdcall</code>	Tells the compiler to use calling convention <code>STDCALL</code> . By default, <code>/iface:stdcall</code> passes arguments by value.
<code>/iface:stdref</code>	Tells the compiler to use calling conventions <code>STDCALL</code> , <code>REFERENCE</code> .

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 Intel® EM64T systems, using `/iface:stdcall` does not change the naming convention.



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

<code>/iface:cvf</code>	Linux and Mac OS: None Windows: <code>/Gm</code>
<code>/iface:mixed_str_len_arg</code>	Linux and Mac OS: <code>-mixed_str_len_arg</code> Windows: None
<code>/iface:nomixed_str_len_arg</code>	Linux and Mac OS: <code>-nomixed_str_len_arg</code> Windows: None
<code>/iface:stdcall</code>	Linux and Mac OS: None Windows: <code>/Gz</code>

See Also

Building Applications: Programming with Mixed Languages Overview and related sections

Intel® Fortran Language Reference: the ATTRIBUTES directive

implicitnone

See [warn.](#)

include

Specifies a directory to add to the include path.

IDE Equivalent

Windows:

General > Additional Include Directories

Preprocessor > Additional Include Directories

Preprocessor>Ignore Standard Include Path (/noinclude)

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: /include:dir
 /noinclude

Arguments

`dir` Is the directory to add to the include path.

Default

OFF The default include path is used.

Description

This option specifies a directory to add to 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 include option for each directory.

To request that the compiler search first in the directory where the source file resides instead of the current directory, specify option `assume source_include`.

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

1. The directory containing the first source file (if `assume source_include` was specified, which is the default).
2. The current default directory where the compilation is taking place.

3. If specified, the directory or directories listed in the include option. The order of searching multiple directories occurs within the specified list from left to right
4. On Linux and Mac OS systems, the directories indicated in the compile-time environment variable FPATH. On Windows, the directories indicated in the environment variable INCLUDE.

On Linux and Mac OS systems, specifying `-noinclude` prevents the compiler from searching in `/usr/include` for files specified in an `INCLUDE` statement. You can specify the `-Idir` option along with this option. This option does not affect the fpp preprocessor behavior, and is not related to the Fortran 95/90 `USE` statement.

On Windows, specifying `/noinclude` (or `/include` or `/I` without a directory, or `/X`) prevents searching in the directory specified by the `INCLUDE` environment variable.

On Windows, specifying the `/noinclude` option negates previous `/include:dir` options.

Alternate Options

`/include` Linux and Mac OS: `-I`
 Windows: `/I`

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

inline

Specifies the level of inline function expansion.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: None

Windows: `/Ob2` `/Ot`

<code>inline size</code>	Linux and Mac OS: None Windows: <code>/Ob2 /Os</code>
<code>inline manual</code>	Linux and Mac OS: None Windows: <code>/Ob0</code>
<code>inline none</code>	Linux and Mac OS: None Windows: <code>/Ob0</code>

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-inline-debug-info`

Mac OS: 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: `-inline_debug_info, -debug inline_debug_info`

Mac OS: None

Windows: `/Qinline_debug_info`

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

OFF 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) 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) or `/Qopt-report` (Windows).



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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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).



When you use this option to change the meaning of inline to "forceinline", the compiler may do so much additional inlining that it runs out of memory and terminates with an "out of memory" message.

Alternate Options

None

See Also

`opt-report`, `Qopt-report` compiler option

Optimizing Applications:

Compiler Directed Inline Expansion of User Functions

Developer Directed Inline Expansion of User Functions

inline-max-per-compile, Qinline-max-per-compile

Specifies the maximum number of times inlining may be applied to an entire compilation unit.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

OFF 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) 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) or `/Qopt-report` (Windows).



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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

OFF 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) 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) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS) or `/Qopt-report` (Windows).



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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

OFF 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) 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) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS) or `/Qopt-report` (Windows).



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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

OFF 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) 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) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS) or `/Qopt-report` (Windows).



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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

OFF 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) 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) or `/Qopt-report` (Windows).

To see compiler values for important inlining limits, specify compiler option `-opt-report` (Linux and Mac OS) or `/Qopt-report` (Windows).



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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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
<code>integer_size 16</code>	Makes default integer and logical variables 2 bytes long. INTEGER and LOGICAL declarations are treated as (KIND=2).
<code>integer_size 32</code>	Makes default integer and logical variables 4 bytes long. INTEGER and LOGICAL declarations are treated as (KIND=4).
<code>integer_size 64</code>	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: `-i2`
Windows: `/4I2`

`integer_size 32` Linux and Mac OS: `-i4`
Windows: `/4I4`

Intel(R) Fortran Compiler Options

`integer_size 64` Linux and Mac OS: `-i8`
 Windows: `/4I8`

ip, Qip

Enables additional interprocedural optimizations for single file compilation.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 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

Linux: `-ip_no_inlining`

Mac OS: None

Windows: `/Qip_no_inlining`

ip-no-pinlining, Qip-no-pinlining

Disables partial inlining enabled by interprocedural optimization options.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T

Syntax

Linux and Mac OS: `-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 systems: `-ip` or `-ipo`
- On Windows systems: `/Qip` or `/Qipo`

It has no effect on other interprocedural optimizations.

To use this option, you must also specify:

- On Linux and Mac OS systems: `-ip` or `-ipo`
- On Windows systems: `/Qip` or `/Qipo`

Alternate Options

Linux: `-ip_no_pinlining`

Mac OS: None

Windows: `/Qip_no_pinlining`

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

Intel® Itanium® architecture

Syntax

Linux: `-IPF-flt-eval-method0`

Mac OS: 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

Linux: `-IPF_flt_eval_method0`

Mac OS: None

Windows: `/QIPF_flt_eval_method0`

IPF-fltacc, QIPF-fltacc

Disables optimizations that affect floating-point accuracy.

IDE Equivalent

Windows: **Floating Point > Floating-Point Accuracy**

Linux: None

Mac OS: None

Architectures

Intel® Itanium® architecture

Syntax

Linux: `-IPF-fltacc`
 `-no-IPF-fltacc`

Mac OS: None

Windows: `/QIPF-fltacc`
 `/QIPF-fltacc-`

Arguments

None

Default

`-no-IPF-fltacc` or Optimizations are enabled that affect floating-point accuracy.
`/QIPF-fltacc-`

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

Linux: `-IPF_fltacc`

Mac OS: None

Windows: `/QIPF_fltacc`

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: None

Architectures

Intel® Itanium® architecture

Syntax

Linux: `-IPF-fma`
 `-no-IPF-fma`

Mac OS: None

Windows: `/QIPF-fma`
 `/QIPF-fma-`

Arguments

None

Default

ON 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 OFF.

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

Linux: `-IPF_fma`

Mac OS: None

Windows: `/QIPF_fma`

See Also

`mp` compiler option

Optimizing Applications: Floating-Point Options for Itanium(R)-Based Systems

IPF-fp-relaxed, QIPF-fp-relaxed

Enables use of faster but slightly less accurate code sequences for math functions.

IDE Equivalent

None

Architectures

Intel® Itanium® architecture

Syntax

Linux: `-IPF-fp-relaxed`
 `-no-IPF-fp-relaxed`

Mac OS: None

Windows: `/QIPF-fp-relaxed`
 `/QIPF-fp-relaxed-`

Arguments

None

Default

`-no-IPF-fp-relaxed` or Default code sequences are used for math functions.
`/QIPF-fp-relaxed-`

Description

This option enables use of faster but slightly less accurate code sequences for math functions, such as divide and sqrt. When compared to strict IEEE* precision, this option slightly reduces the accuracy of floating-point calculations performed by these functions, usually limited to the least significant digit.

This option also enables the performance of more aggressive floating-point transformations, which may affect accuracy.

Alternate Options

Linux: `-IPF_fp_relaxed`
Mac OS: None
Windows: `/QIPF_fp_relaxed`

IPF-fp-speculation, QIPF-fp-speculation

Tells the compiler the mode to speculate on floating-point (FP) operations.

IDE Equivalent

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

Linux: None

Mac OS: None

Architectures

Intel® Itanium® architecture

Syntax

Linux: `-IPF-fp-speculationmode`

Mac OS: None

Windows: `/QIPF-fp-speculationmode`

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 speculate on floating-point operations only when safe.
- `strict` Tells the compiler to disable speculation on floating-point operations.
- `off` Same as strict.

Default

<code>-IPF-fp-speculationfast</code> or <code>/QIPF-fp-speculationfast</code>	The compiler speculates on floating-point operations when optimizations are enabled. If you specify no optimizations (<code>-O0</code> on Linux; <code>/Od</code> on Windows), the default is <code>-IPF-fp-speculationsafe</code> (Linux) or <code>/QIPF-fp-speculationsafe</code> (Windows).
---	---

Description

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

Alternate Options

Linux: `-IPF_fp_speculation`

Mac OS: None

Windows: `/QIPF_fp_speculation`

See Also

Optimizing Applications: Floating-Point Options for Itanium(R)-Based Systems

ipo, Qipo

Enables interprocedural optimizations between files.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 Optimizations Overview
Generating Multiple IPO Object Files
IPO Compilation Model

ipo-c, Qipo-c

Tells the compiler to optimize across multiple files and generate a single object file.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 systems; `ipo_out.obj` on Windows systems).

It performs the same optimizations as `-ipo` (Linux and Mac OS) 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

Linux: `-ipo_c`

Mac OS: None

Windows: `/Qipo_c`

See Also

`ipo`, `Qipo` compiler option

ipo-S, Qipo-S

Tells the compiler to optimize across multiple files and generate a single assembly file.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 systems; `ipo_out.asm` on Windows systems).

It performs the same optimizations as `-ipo` (Linux) 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

Linux: `-ipo_S`

Mac OS: None

Windows: `/Qipo_S`

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-ipo-separate`

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

Linux: `-ipo_separate`

Mac OS: None

Windows: `/Qipo_separate`

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-isystemdir`

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: None

Architectures

Intel® Itanium® architecture

Syntax

Linux: `-ivdep-parallel`

Mac OS: 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

Linux: `-ivdep_parallel`

Mac OS: None

Windows: `/Qivdep_parallel`

See Also

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

Kpic

See [fpic](#).

I

Tells the linker to search for a specified library when linking.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-lstring`

Windows: None

Arguments

string Specifies the library (*libstring*) 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-Ldir`

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: <ul style="list-style-type: none">• 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: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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	<code>/libs:static</code> or <code>/ML</code> <code>/libs</code> or <code>/static</code>	
Multithreaded	<code>/libs:static</code> <code>/threads</code>	<code>/MT</code>
Debug single-threaded	<code>/libs:static</code> <code>/dbglibs</code>	<code>/MLd</code>
Debug multithreaded	<code>/libs:static</code> <code>/threads</code> <code>/dbglibs</code>	<code>/MTd</code>
Single-threaded, dynamic-link libraries (DLLs)	<code>/libs:dll</code>	<code>/MDs</code>
Debug single-threaded, dynamic-link libraries (DLLs)	<code>/libs:dll</code> <code>/dbglibs</code>	<code>/MDsd</code>
Multithreaded DLLs	<code>/libs:dll</code> <code>/threads</code>	<code>/MD</code>
Multithreaded debug DLLs	<code>/libs:dll</code> <code>/threads</code> <code>/dbglibs</code>	<code>/MDd</code>
Fortran QuickWin multi-doc applications	<code>/libs:qwin</code>	<code>/MW</code>
Fortran standard graphics (QuickWin single-doc) applications	<code>/libs:qwins</code>	<code>/MWs</code>
Debug Fortran QuickWin multi-doc applications	<code>/libs:qwin</code> <code>/dbglibs</code>	None
Debug Fortran standard graphics (QuickWin single-doc) applications	<code>/libs:qwins</code> <code>/dbglibs</code>	None

Alternate Options

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

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

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

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

See Also

Intel(R) Fortran Compiler Options

`threads` compiler option

`dbglibs` compiler option

Building Applications:
Programming with Mixed Languages Overview

link

Passes user-specified options directly to the linker at compile time.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -logo
-nologo

Windows: /logo
/nologo

Arguments

None

Default

Linux and Mac OS: OFF The compiler version information is not displayed.

Windows: ON 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: -v

Windows: None

lowercase

See [names](#).

map

Tells the linker to generate a link map file.

IDE Equivalent

Windows: **General > Suppress Startup Banner**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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

OFF 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-map-opts`

Mac OS: 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 an IA-32 processor, you will only see equivalent options that are available on IA-32 processors.

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: Option Mapping Tool

mcmmodel

Tells the compiler to use a specific memory model to generate code and store data.

IDE Equivalent

None

Architectures

Intel® EM64T

Syntax

Linux: `-mcmmodel=mem_model`

Mac OS: 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

`mcmmodel=small` 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, `-mcmmodel=small` is sufficient. COMMONs larger than 2GB require `-mcmmodel=medium` or `-mcmmodel=large`. Allocation of memory larger than 2GB can be done with any setting of `-mcmmodel`.

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.



When you specify `-mcmodel=medium` or `-mcmodel=large`, you must also specify compiler option `-i-dynamic` 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 the 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

`i-dynamic` compiler option

`fpic` compiler option

MD

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: None

Windows: `/libs:dll`

See Also

`libs` 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: /ML
 /MLd

Arguments

None

Default

Intel EM64T systems: OFF	On Intel EM64T systems, the linker searches for unresolved references in a multithreaded, static run-time library. On IA-32 and Intel Itanium systems, the linker searches for unresolved references in a single-threaded, static run-time library.
IA-32 and Intel Itanium systems: ON	

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: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

Intel(R) Fortran Compiler Options

`fltconsistency` compiler option
`mp` compiler option

mrelax

Tells the compiler to pass linker option `-relax` to the linker.

IDE Equivalent

None

Architectures

Intel® Itanium® architecture

Syntax

Linux: `-mrelax`
 `-mno-relax`

Mac OS: None

Windows: None

Arguments

None

Default

OFF 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

MT

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: /MT
 /MTd

Arguments

None

Default

Intel EM64T systems: ON	On Intel EM64T systems, the linker searches for unresolved references in a multithreaded, static run-time library. On IA-32 and Intel Itanium systems, the linker searches for unresolved references in a single-threaded, static run-time library.
IA-32 and Intel Itanium systems: OFF	

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`. You can also specify `/MTd`, where *d* indicates a debug version.

Alternate Options

None

See Also

`libs` compiler option

`threads` compiler option

mtune

Performs optimizations for a specified CPU.

IDE Equivalent

None

Architectures

IA-32, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-mtune=processor`

Windows: None

Arguments

processor Is the CPU. Possible values on IA-32 systems are:

<code>pentium</code>	Optimizes for Intel® Pentium® processors.
<code>pentiumpro</code>	Optimizes for Intel® Pentium® Pro, Intel Pentium II, and Intel Pentium III processors.
<code>pentium4</code>	Optimizes for Intel® Pentium® 4 processors.
<code>pentium-mmx</code>	Optimizes for Intel® Pentium® with MMX™ technology.

Possible values on Itanium®-based systems are:

<code>itanium</code>	Optimizes for Intel® Itanium® processors.
<code>itanium2</code>	Optimizes for Intel® Itanium® 2 processors.
<code>itanium2-p9000</code>	Optimizes for Dual-Core Intel® Itanium® 2 Processor 9000 Sequence processors. 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 Sequence processors.

Default

`pentium4` On IA-32 systems, the compiler optimizes for Intel Pentium 4 processors.

`itanium2` On Itanium®-based systems, the compiler optimizes for Intel® Itanium® 2 processors.

Description

This option performs optimizations for a specified CPU.

Alternate Options

mtune=itanium2-p9000 Linux and Mac OS: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 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.



On Windows systems, if you specify option `/iface:ceref`, 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:ceref` and want external names to be uppercase, you must explicitly specify option `/names:uppercase`.

Alternate Options

`names lowercase` Linux and Mac OS: `-lowercase`
Windows: `/Qlowercase`

`names uppercase` Linux and Mac OS: `-uppercase`
Windows: `/Quppercase`

See Also

`iface` compiler option

Intel® Fortran Language Reference: the ALIAS directive

nbs

See [assume](#).

no-cpprt

See [cxxlib](#).

nobss-init, Qnobss-init

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-nobss-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: `-nobss_init`

Mac OS: None

Windows: `/Qnobss_init`

nodefaultlibs

Prevents the compiler from using standard libraries when linking.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

nostartfiles

Prevents the compiler from using standard startup files when linking.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-o \textit{file}`

Windows: None

Arguments

file is the name for the output file.

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.

Alternate Options

Linux and Mac OS: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -O [*n*]

Windows: /O [*n*]

Arguments

n Is the optimization level. Possible values are 1, 2, or 3. On Linux and Mac OS 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)	This is the same as specifying O2.
O0 (Linux and Mac OS)	Disables all optimizations. On IA-32 and Intel® EM64T systems, this option sets option -fp 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 Itanium®-based systems, it disables software pipelining, loop unrolling, and global code scheduling.

On Intel® Itanium® processors, 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 systems:
`-unroll0, -nofltconsistency` (same as `-mno-ieee-fp`), `-fp`
 (same as `-fomit-frame-pointer`)
- On IA-32 Windows systems:
`/unroll0` (or `/Qunroll0`), `/nofltconsistency` (same as `/Op-`),
`/Oy`, `/Os`, `/Ob2`
- On Intel® EM64T and Itanium®-based Windows systems:
`/unroll0` (or `/Qunroll0`), `/nofltconsistency` (same as `/Op-`),
`/Os`, `/Ob2`

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 Itanium-based systems, this option enables optimizations for speed, including global code scheduling, software pipelining, predication, and speculation.

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 Linux and Mac OS systems, this option is the same as the `○` option.

On Windows systems, this option is the same as the `○x` option.

The `○2` option sets the following options:

- On Windows IA-32 systems:
`/Og`, `/Ot`, `/Oy`, `/Ob2`, and `/Gs`.
- On Intel EM64T Windows systems:
`/Og`, `/Ot`, `/Ob2`, and `/Gs`.

On Linux and Mac OS systems, if `-g` is specified, `○2` is turned off and `○0` is the default unless `○2` (or `○1` or `○3`) is explicitly specified in the command line together with `-g`.

`○3`

Enables `○2` 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 `○3` option sets the `/Ob2` option.

On Linux and Mac OS systems, the `○3` option sets the `-fp` option.

On IA-32 and Intel EM64T processors, when `○3` 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 `○2`, which may result in longer compilation times.

On Intel Itanium processors, the `○3` option enables optimizations for technical computing applications (loop-intensive code): loop optimizations and data prefetch.

The `○3` 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 `○2` optimizations.

The `○3` option is recommended for applications that have loops that heavily use floating-point calculations and process large data sets.

The last `○` 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: None
Windows: `/Od, /optimize:0, /nooptimize`
- `O1` Linux and Mac OS: None
Windows: `/optimize:1, /optimize:2`
- `O2` Linux and Mac OS: None
Windows: `/Ox, /optimize:3, /optimize:4`
- `O3` Linux and Mac OS: None
Windows: `/optimize:5`

See Also

`Od` compiler option

`fast` compiler option

Optimizing Applications:
 Compiler Optimizations Overview
 Optimization Options Summary
 Efficient Compilation

Ob

Specifies the level of inline function expansion.

IDE Equivalent

Windows: **Optimization > Inline Function Expansion**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-Obn`

Windows: `/Obn`

Arguments

n Is the inline function expansion level. Possible values are 0, 1, and 2.

Default

`Ob2` If option `Ob2` is specified or is in effect by default.

`Ob0` 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
<code>Ob0</code>	Disables inlining of user-defined functions. Note that statement functions are always inlined.
<code>Ob1</code>	Enables inlining when an inline keyword or an inline directive is specified.
<code>Ob2</code>	Enables inlining of any function at the compiler's discretion.

Alternate Options

None

See Also

`inline` compiler option

object

Specifies the name for an object file.

IDE Equivalent

Windows: **Output Files > Object File Name**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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 systems, this option is equivalent to specifying option `-ofile -c`.

Alternate Options

Linux and Mac OS: None

Windows: `/Fo`

See Also

- compiler option

Od

Disables all optimizations.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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 systems, this option sets the /Oy- option.

Alternate Options

Linux and Mac OS: -O0

Windows: /optimize:0

See Also

o compiler option

Og

Enables global optimizations.

IDE Equivalent

Windows: **Optimization > Global Optimizations**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: /Og

Arguments

None

Default

ON 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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: `-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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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.

Alternate Options

None

See Also

`openmp_stubs` compiler option

openmp-profile, Qopenmp-profile

Enables analysis of OpenMP* applications if Intel® Thread Profiler is installed.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-openmp-profile`

Mac OS: 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. If this threading tool is not installed, this option has no effect.

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

Linux: `-openmp_profile`

Mac OS: None

Windows: `/Qopenmp_profile`

openmp-report, Qopenmp-report

Controls the OpenMP* parallelizer's level of diagnostic messages.

IDE Equivalent

Windows: **Diagnostics > OpenMP Diagnostic Level**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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` 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.
or `/Qopenmp-report1`

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) 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

Linux: `-openmp_report`

Mac OS: None

Windows: `/Qopenmp_report`

See Also

Optimizing Applications:

Parallelization with OpenMP* Overview

Compiling with OpenMP, Directive Format, and Diagnostics

openmp-stubs, Qopenmp-stubs

Enables compilation of OpenMP programs in sequential mode.

IDE Equivalent

Windows: **Language > Process OpenMP Directives**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

Linux: `-openmp_stubs`

Mac OS: None

Windows: `/Qopenmp_stubs`

See Also

`openmp`, `Qopenmp` compiler options

opt-mem-bandwidth, Qopt-mem-bandwidth

Enables performance tuning and heuristics that control memory bandwidth use among processors.

IDE Equivalent

None

Architectures

Intel® Itanium® architecture

Syntax

Linux: `-opt-mem-bandwidthn`

Mac OS: None

Windows: `/Qopt-mem-bandwidthn`

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

<code>-opt-mem-bandwidth⁰</code> or <code>/Qopt-mem-bandwidth⁰</code>	For serial (non-parallel) compilation, a set of performance tuning and heuristics in compiler optimizations is enabled that is optimal for serial code.
<code>-opt-mem-bandwidth¹</code> or <code>/Qopt-mem-bandwidth¹</code>	If you specify compiler option <code>-parallel</code> (Linux) or <code>/Qparallel</code> (Windows), <code>-openmp</code> (Linux) or <code>/Qopenmp</code> (Windows), or Cluster OpenMP option <code>-cluster-openmp</code> , 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-report, Qopt-report

Tells the compiler to generate an optimization report to `stderr`.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-opt-report`

Windows: `/Qopt-report`

Arguments

None

Default

OFF No optimization report is generated.

Description

This option tells the compiler to generate an optimization report to `stderr`.

Alternate Options

Linux: `-opt_report`

Mac OS: None

Windows: `/Qopt_report`

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-opt-report-filefile`

Windows: `/Qopt-report-filefile`

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) or `/Qopt-report` (Windows).

Alternate Options

Linux: `-opt_report_file`

Mac OS: None

Windows: `/Qopt_report_file`

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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) or `/Qopt-report-phase` (Windows). No compilation is performed.

Alternate Options

Linux: `-opt_report_help`

Mac OS: None

Windows: `/Qopt_report_help`

See Also

`opt-report`, `Qopt-report` compiler options

`opt-report-phase`, `Qopt-report-phase` compiler options

Optimizing Applications: Optimizer Report Generation

opt-report-level, Qopt-report-level

Specifies the detail level of the optimization report.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-opt-report-level [level]`

Windows: `/Qopt-report-level [level]`

Arguments

level is the detail level. Possible values are min, med, or max.

Default

`-opt-report-levelmin` or Minimal details appear in the optimization report.
`/Qopt-report-levelmin`

Description

This option specifies the detail level of the optimization report.

Alternate Options

Linux: `-opt_report_level`

Mac OS: None

Windows: `/Qopt_report_level`

See Also

`opt-report`, `Qopt-report` compiler options

Optimizing Applications: Optimizer Report Generation

opt-report-phase, Qopt-report-phase

Specifies the optimizer phase to use when reports are generated.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-opt-report-phase $phase$`

Windows: `/Qopt-report-phase $phase$`

Arguments

phase is the phase to generate reports for. Possible values are:

<code>ipo</code>	Interprocedural Optimizer
<code>hlo</code>	High Level Optimizer
<code>ilo</code>	Intermediate Language Scalar Optimizer
<code>ecg</code>	Code Generator (Windows and Linux systems on Intel Itanium processors only)
<code>ecg_swp</code>	Software pipelining component of the Code Generator (Windows and Linux systems on Intel Itanium processors only)
<code>pgo</code>	Profile Guided Optimization
<code>all</code>	All phases

Default

OFF No optimization reports are generated.

Description

This option specifies the optimizer phase to use when reports are generated. To use this option, you must also specify `-opt-report` (Linux and Mac OS) 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.

Alternate Options

Linux: `-opt_report_phase`

Mac OS: None

Windows: `/Qopt_report_phase`

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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.

If this option is used, but no *string* is specified, reports from all routines are generated.

Alternate Options

Linux: `-opt_report_routine`

Mac OS: None

Windows: `/Qopt_report_routine`

See Also

`opt-report`, `Qopt-report` compiler options

Optimizing Applications: Optimizer Report Generation

optimize

See [o](#).

Os

Enables most speed optimizations.

IDE Equivalent

Windows: **Optimization > Favor Size or Speed**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: /Ot

Arguments

None

Default

ON Optimizations are made for code speed.

If o0 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 [fp](#), [Oy](#).

p

Compiles and links for function profiling with gprof(1).

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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: `-qp`, `-pg` (not available on Itanium®-based systems)

Windows: None

P

See [preprocess_only](#).

pad, Qpad

Enables the changing of the variable and array memory layout.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

Windows: `/Qpad`
`/Qpad-`

Arguments

None

Default

OFF 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

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

Arguments

None

Default

OFF 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, `nopad-source`, 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) or `/warn:nousage` (Windows).

Specifying `pad-source` can prevent warning messages associated with option `warn-usage`.

Alternate Options

Linux: `-pad_source`

Mac OS: None

Windows: `/pad_source`, `/Qpad_source`

See Also

`warn` compiler option

par-report, Qpar-report

Controls the autoparallelizer's level of diagnostic messages.

IDE Equivalent

Windows: **Diagnostics > Auto-Parallelizer Diagnostic Level**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-par-report [n]`

Windows: `/Qpar-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 successfully auto-parallelized. The compiler also issues a "LOOP AUTO-PARALLELIZED" message for parallel loops.
- ² Diagnostic messages are displayed indicating loops successfully and unsuccessfully auto-parallelized.
- ³ The same diagnostic messages are displayed as 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 indicating loops successfully auto-parallelized. If you do not specify the option on the command line, the default is to display no parallel diagnostic messages.

Description

This option controls the autoparallelizer's level of diagnostic messages. To use this option, you must also specify `-parallel` (Linux and Mac OS) 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

Linux: `-par_report`

Mac OS: None

Windows: `/Qpar_report`

See Also

Optimizing Applications:

Auto-Parallelization Overview

Auto-Parallelization: Enabling, Options, Directives, and Environment Variables

Auto-Parallelization: Threshold Control and Diagnostics

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

<code>-par-</code>	Loops get auto-parallelized only if profitable parallel execution is
<code>threshold100</code> or	almost certain. This is also the default if you do not specify <i>n</i> .
<code>/Qpar-</code>	
<code>threshold100</code>	

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) 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

Linux: `-par_threshold`

Mac OS: None

Windows: `/Qpar_threshold`

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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`.

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, Intel® EM64T

Syntax

Linux and Mac OS: `-pcn`

Windows: `/Qpcn`

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` On Linux* and Mac OS* systems, the floating-point significand is rounded to 64 bits. On Windows* systems, the floating-point significand is rounded to 53 bits.
or
`/Qpc64`

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) or `/Qpc32` (Windows) option or by user intervention, may affect the results returned by some of the mathematical functions.

Alternate Options

None

See Also

pdbfile

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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, Intel® EM64T

Syntax

Linux and Mac OS: `-prec-div`
`-no-prec-div`

Windows: `/Qprec-div`
`/Qprec-div-`

Arguments

None

Default

ON 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) or `/Qprec-div-` (Windows), it enables optimizations that give slightly less precise results than full IEEE division.

Alternate Options

Linux: `-prec_div`

Mac OS: None

Windows: `/Qprec_div`

See Also

Optimizing Applications: Floating-point Options for IA-32 and Intel(R) EM64T

prec-sqrt, Qprec-sqrt

Improves precision of square root implementations.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T

Syntax

Linux and Mac OS: `-prec-sqrt`
`-no-prec-sqrt`

Windows: `/Qprec-sqrt`
`/Qprec-sqrt-`

Arguments

None

Default

OFF The compiler uses a faster but less precise implementation of square root.

Note that the default is ON 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 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: None

Architectures

Intel® Itanium® architecture

Syntax

Linux: `-prefetch`
 `-no-prefetch`

Mac OS: None

Windows: `/Qprefetch`
 `/Qprefetch-`

Arguments

None

Default

ON Prefetch insertion optimization is enabled.

Description

This option enables prefetch insertion optimization. To use this option, you must also specify `O3`.

The goal of prefetching is to reduce cache misses by providing hints to the processor about when data should be loaded into the cache.

To disable the prefetch insertion optimization, use `-no-prefetch` (Linux) or `/Qprefetch-` (Windows).

Alternate Options

None

See Also

Optimizing Applications: Coding Guidelines for Intel(R) Architectures

preprocess_only

Causes the Fortran preprocessor to send output to a file.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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: `-P`, `-F` (this is a deprecated option)

Mac OS: `-P`

Windows: `/P`

print-multi-lib

Prints information about where system libraries should be found.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-print-multi-lib`

Windows: None

Arguments

None

Default

OFF No information is printed unless the option is specified.

Description

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

Linux: `-prof_dir`

Mac OS: None

Windows: `/Qprof_dir`

See Also

Optimizing Applications:
Advanced PGO Options
Coding Guidelines for Intel(R) Architectures

prof-file, Qprof-file

Specifies a file name for the profiling summary file.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-prof-file file`

Windows: `/Qprof-file file`

Arguments

file is the name of the profiling summary file.

Default

OFF The profiling summary file has the name of the source file with extension `.dyn` or `.dpi`.

Description

This option specifies a file name for the profiling summary file.

Alternate Options

Linux: `-prof_file`

Mac OS: None

Windows: `/Qprof_file`

See Also

Optimizing Applications:
 Advanced PGO Options
 Coding Guidelines for Intel(R) Architectures
 Example of Profile-Guided Optimization

prof-format-32, Qprof-format-32

Produces profile data with 32-bit counters.
This option has been deprecated.

IDE Equivalent

None

Architectures

IA-32, Intel® Itanium® architecture

Syntax

Linux: `-prof-format-32`

Mac OS: None

Windows: `/Qprof-format-32`

Arguments

None

Default

OFF The compiler produces profile data with 64-bit counters to handle large numbers of events.

Description

This option produces profile data with 32-bit counters. It allows compatibility with earlier compilers.

Alternate Options

Linux: `-prof_format_32`

Mac OS: None

Windows: `/Qprof_format_32`

prof-gen, Qprof-gen

Instruments a program for profiling.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-prof-gen`

Windows: `/Qprof-gen`

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).

It is 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

Linux: `-prof_gen`

Mac OS: None

Windows: `/Qprof_gen`

See Also

`prof-genx`, `Qprof-genx` compiler options

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

Syntax

Linux and Mac OS: `-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) 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

Linux: `-prof_gen_sampling`

Mac OS: None

Windows: `/Qprof_gen_sampling`

See Also

`prof-use`, `Qprof-use` compiler options

`ssp`, `Qssp` compiler options

prof-genx, Qprof-genx

Instruments a program for profiling and gathers extra information for code coverage tools.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-prof-genx`

Windows: `/Qprof-genx`

Arguments

None

Default

OFF Programs are not instrumented for profiling.

Description

This option instruments a program for profiling and gathers extra information (source position) for code coverage tools.

Like `-prof-gen` (Linux) or `/Qprof-gen` (Windows), it is 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. It also creates a new static profile information file (.spi).

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.

Alternate Options

Linux: `-prof_genx`

Mac OS: None

Windows: `/Qprof_genx`

See Also

`prof-gen`, `Qprof-gen` compiler options

prof-use, Qprof-use

Enables the use of profiling information during optimization.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

Linux: `-prof_use`

Mac OS: None

Windows: `/Qprof_use`

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: None

Architectures

Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: /Qchkstk
 /Qchkstk-

Arguments

None

Default

ON 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](#).

Qdps

See [altparam](#).

Qdyncom

See [dyncom](#), [Qdyncom](#).

Qextend_source

See [extend_source](#).

Qfp-port

See [fp-port](#), [Qfp-port](#).

Qfnsplit

See [fnsplit](#), [Qfnsplit](#).

Qfpp

See [fpp](#), [Qfpp](#).

Qfpstkchk

See [fpstkchk](#), [Qfpstkchk](#).

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: None

Architectures

Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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

Linux and Mac OS: None

Windows: /QIA64_fr32

Qlfist

See [rcd](#), [Qrcd](#).

Qinline-debug-info

See [inline-debug-info](#), [Qinline-debug-info](#).

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

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-S

See [ipo-S](#), [Qipo-S](#).

Qipo-separate

See [ipo-separate](#), [Qipo-separate](#).

Qivdep-parallel

See [ivdep-parallel](#), [Qivdep-parallel](#).

Qlocation

Specifies the directory for supporting tools.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 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 [nobss-init](#), [Qnobss-init](#).

Qonetrip

See [onetrip](#), [Qonetrip](#).

Qopenmp

See [openmp](#), [Qopenmp](#).

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-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-level](#), [Qopt-report-level](#).

Qopt-report-phase

See [opt-report-phase](#), [Qopt-report-phase](#).

Qopt-report-routine

See [opt-report-routine](#), [Qopt-report-routine](#).

Qoption

Passes options to a specified tool.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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. To use this option, you must also specify `-ip` or `-ipo` (Linux and Mac OS), or `/Qip` or `/Qipo` (Windows).

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:

- `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 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 systems:

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

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

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 command activates procedural and interprocedural optimizations on source.f and sets the maximum increase in the number of intermediate language statements to 5 for each function:

On Windows systems:

```
ifort /Qip /Qoption,f,-ip_ninl_max_stats=5 source.f
```

On Linux and Mac OS systems:

```
ifort -ip -Qoption,f,-ip_ninl_max_stats=5 source.f
```

See Also

Qlocation compiler option

qp

See [p](#).

Qpad

See [pad](#), [Qpad](#).

Qpad-source

See [pad-source](#), [Qpad-source](#).

Qpar-report

See [par-report](#), [Qpar-report](#).

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-format-32

See [prof-format-32](#), [Qprof-format-32](#).

Qprof-gen

See [prof-gen](#), [Qprof-gen](#).

Qprof-gen-sampling

See [prof-gen-sampling](#), [Qprof-gen-sampling](#).

Qprof-genx

See [prof-genx](#), [Qprof-genx](#).

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

Syntax

Linux and Mac OS: 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

None

Qsafe-cray-ptr

See [safe-cray-ptr](#), [Qsafe-cray-ptr](#).

Qsave

See [save](#), [Qsave](#).

Qscalar-rep

See [scalar-rep](#), [Qscalar-rep](#).

Qsfalign

Specifies stack alignment for functions.

IDE Equivalent

None

Architectures

IA-32

Syntax

Linux: 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.
- ¹⁶ Specifies that alignment should occur for functions with 16-byte aligned variables.

Default

/Qsfalign⁸ Alignment occurs for functions with 8-byte aligned variables.

Description

This option specifies stack alignment for functions.

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](#).

Qtrapuv

See [ftrapuv](#), [Qtrapuv](#).

Qunroll

See [unroll](#), [Qunroll](#).

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

Syntax

Linux and Mac OS: 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

Qvc

Specifies compatibility with Microsoft* Visual C++ or Microsoft* Visual Studio.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T

Syntax

Linux and Mac OS: None

Windows: /Qvc6
 /Qvc7
 /Qvc7.1
 /Qvc8

Arguments

None

Default

OFF No compatibility with Microsoft Visual C++ or Visual Studio is enabled.

However, on IA-32 systems, if Visual C++ 6.0, Visual Studio .NET 2002, or Visual Studio .NET 2003 is installed, compatibility is enabled by default.

Description

This option specifies compatibility with Visual C++ or Visual Studio.

Option	Description
/Qvc6	Specifies compatibility with Visual C++ 6.0.
/Qvc7	Specifies compatibility with Microsoft* Visual Studio .NET 2002.
/Qvc7.1	Specifies compatibility with Microsoft* Visual Studio .NET 2003.
/Qvc8	Specifies compatibility with Microsoft* Visual Studio 2005.

On Intel® EM64T systems, /Qvc7.1 is the only valid option.

Alternate Options

None

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, Intel® EM64T

Syntax

Linux and Mac OS: `-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: None

Windows: `/QIfist`

real_size

Specifies the default KIND for real variables.

IDE Equivalent

Windows: **Data > Default Real KIND**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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
<code>real_size 32</code>	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)).
<code>real_size 64</code>	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)).
<code>real_size 128</code>	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: `-r8, -autodouble`
Windows: `/4R8, /Qautodouble`

`real_size 128` Linux and Mac OS: `-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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Systems

Windows, Linux

Syntax

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

Windows: `/recursive`
`/norecursive`

Arguments

None

Default

OFF 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

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

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

Arguments

keyword Specifies details about the program. Possible values are:

- | | |
|-----------------------|--|
| <code>none</code> | 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 <code>noreentrancy</code> . |
| <code>async</code> | 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. |
| <code>threaded</code> | 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

OFF 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 keyword `uninit` in [check](#).

S

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 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: 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

Linux: `-safe_cray_ptr`

Mac OS: None

Windows: `/Qsafe_cray_ptr`

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) 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-save`

Windows: `/Qsave`

Arguments

None

Default

`-auto_scalar` Scalar variables of intrinsic types INTEGER, REAL, COMPLEX, and LOGICAL are allocated to the run-time stack. Note that if option `/Qauto_scalar` recursive, `-openmp` (Linux and Mac OS), 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 on the run-time stack, specify option `automatic`.

Alternate Options

Linux and Mac OS: `-noautomatic`, `-noauto`

Windows: `/noautomatic`, `/noauto`, `/4Na`

See Also

`automatic` compiler option

`auto_scalar` compiler option

scalar-rep, Qscalar-rep

Enables scalar replacement performed during loop transformation.

IDE Equivalent

None

Architectures

IA-32

Syntax

Linux and Mac OS: `-scalar-rep`
`-no-scalar-rep`

Windows: `/Qscalar-rep`
`/Qscalar-rep-`

Arguments

None

Default

OFF 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

Linux: `-scalar_rep`
 Mac OS: None
 Windows: `/Qscalar_rep`

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-shared`

Mac OS: 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 IA-32 systems and Intel® EM64T systems, 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-libcxa

Links the Intel `libcxa` C++ library dynamically.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-shared-libcxa`

Mac OS: None

Windows: None

Arguments

None

Default

ON 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

source

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: `-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-sox`
`-no-sox`

Windows: `/Qsox`
`/Qsox-`

Arguments

None

Default

OFF 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 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.

Alternate Options

None

ssp, Qssp

Enables Software-based Speculative Pre-computation (SSP) optimization.

IDE Equivalent

None

Architectures

IA-32

Syntax

Linux: `-ssp`

Mac OS: 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)" in your Optimizing Guide.

Alternate Options

None

See Also

Optimizing Applications: SSP Precomputation (IA-32)

stand

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

IDE Equivalent

Windows: **Diagnostics > Warn For Nonstandard Fortran**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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.

Default

OFF 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 95`.

Option	Description
<code>stand</code> <code>none</code>	Causes the compiler to issue no messages for nonstandard language elements. This is the same as specifying <code>nostand</code> .
<code>stand</code>	Causes the compiler to issue messages for language elements that are not

Intel(R) Fortran Compiler Options

`f90` standard in Fortran 90.

`stand` Causes the compiler to issue messages for language elements that are not
`f95` standard in Fortran 95. This option is set if you specify `warn stderrs`.

Alternate Options

`stand none` Linux and Mac OS: `-nostand`
Windows: `/nostand`

`stand f90` Linux and Mac OS: `-std90`
Windows: None

`stand f95` Linux and Mac OS: `-stand, -std95, -std`
Windows: `/stand`

See Also

`warn` compiler option

static

Prevents linking with shared libraries.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-static`

Mac OS: None

Windows: `/static`

Arguments

None

Default

ON 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-libcxa

Links the Intel `libcxa` C++ library statically.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-static-libcxa`

Mac OS: 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

std, std90, std 95

See [stand](#).

syntax

This option has been deprecated. See [syntax-only](#).

syntax-only

Tells the compiler to check only for correct syntax.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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: `-syntax_only`, `-y`, `-fsyntax-only`, `-syntax` (this is a deprecated option)

Mac OS: `-y`, `-fsyntax-only`

Windows: `/Zs`, `/syntax_only`

T

Tells the linker to read link commands from a file.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: -T *file*

Mac OS: 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux: `-tcheck`

Mac OS: None

Windows: `/Qtcheck`

Arguments

None

Default

OFF Threaded applications are not analyzed.

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` 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

Tf

See [source](#).

threads

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-threads`
`-nothreads`

Windows: `/threads`
`/nothreads`

Arguments

None

Default

Intel EM64T systems: ON	On IA-32 and Intel Itanium systems, the linker does not search for unresolved references in a multithreaded run-time library. On Intel EM64T systems, it does.
IA-32 and Intel Itanium systems: OFF	

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	<code>/libs:static</code> <code>/threads</code>	<code>/MT</code>
Debug multithreaded	<code>/libs:static</code> <code>/threads</code> <code>/dbglibs</code>	<code>/MTd</code>

Intel(R) Fortran Compiler Options

Multithreaded DLLs	<code>/libs:dll</code> <code>/threads</code>	<code>/MD</code>
Multithreaded debug DLLs	<code>/libs:dll</code> <code>/threads</code> <code>/dbglibs</code>	<code>/MDd</code>

Alternate Options

None

See Also

Building Applications: Programming with Mixed Languages Overview

tpp1, tpp2, G1, G2, G2-p9000

Optimizes application performance for Intel® Itanium® processors.

IDE Equivalent

Windows: **Optimization > Optimize For Intel® Processor**

Linux: None

Mac OS: None

Architectures

Intel® Itanium® architecture

Syntax

Linux: -tpp1
 -tpp2

Mac OS: None

Windows: /G1
 /G2
 /G2-p9000

Arguments

None

Default

-tpp2 or /G2 Performance is optimized for Intel® Itanium® 2 processors.

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 Itanium architecture.

Option	Description
tpp1 or G1	Optimizes for Intel® Itanium® processors.
tpp2 or G2	Optimizes for Intel® Itanium® 2 processors.
G2- p9000	Optimizes for Dual-Core Intel® Itanium® 2 Processor 9000 Sequence processors. 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 Sequence processors.

These options always generate code that is backwards compatible with Intel processors of the same architecture. For example, code generated with option `tpp2` (Linux) or `G2` (Windows) runs correctly on Itanium and Itanium 2 processors, although performance may be faster on Itanium processors when compiled using `tpp1` or `G1`.

Alternate Options

G2-p9000 Linux: `mtune=itanium2-p9000`
Mac OS: 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 Itanium 2 processor by default. The same binary will also run on Itanium processors. All lines in the code example are equivalent.

```
ifort prog.f
ifort -tpp2 prog.f      ! command on Linux
ifort /G2 prog.f        ! command on Windows
```

In the following example, the compiled binary is optimized for the Itanium processor:

```
ifort -tpp1 prog.f      ! command on Linux
ifort /G1 prog.f        ! command on Windows
```

tpp5, tpp6, tpp7, G5, G6, G7

Optimize application performance for IA-32 and Intel® EM64T processors.
These options have been deprecated.

IDE Equivalent

Windows: **Optimization > Optimize For Intel(R) Processor** (/GB, /G5, /G6, /G7)

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T

Syntax

Linux: -tpp5
 -tpp6
 -tpp7

Mac OS: None

Windows: /G5
 /G6
 /G7

Arguments

None

Default

-tpp7 On IA-32 and Intel® EM64T systems, performance is optimized for Intel®
or /G7 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
tpp5 or G5	Optimizes for Intel® Pentium® and Pentium® with MMX™ technology processors.
tpp6	Optimizes for Intel® Pentium® Pro, Pentium® II and Pentium® III processors.

G6

`tpp7` or `G7` Optimizes for Intel® Core™ Duo processors, Intel® Core™ Solo processors, Intel® Pentium® 4 processors, Intel® Xeon® processors, Intel® Pentium® M processors, and Intel® Pentium® 4 processors with Streaming SIMD Extensions 3 (SSE3) instruction support.

On Intel® EM64T systems, only option `tpp7` (Linux) or `G7` (Windows) is valid.

These options always generate code that is backwards compatible with Intel processors of the same architecture. For example, code generated with the `tpp7` or `G7` option runs correctly on Pentium III processors, although performance may be faster on Pentium III processors when compiled using `tpp6` (Linux) or `G6` (Windows).

On Linux IA-32 systems, it is recommended you use option `-mtune` instead of the `-tpp` options.

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 -tpp7 prog.f      ! command on Linux
ifort /G7 prog.f        ! command on Windows
```

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

```
ifort -tpp5 prog.f      ! command on Linux
ifort /G5 prog.f        ! command on Windows
```

See Also

`mtune` compiler option

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-traceback`
`-notraceback`

Windows: `/traceback`
`/notraceback`

Arguments

None

Default

OFF 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.

Intel(R) Fortran Compiler Options

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 IA-32 and Intel® EM64T processors, 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 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: Obtaining Traceback Information with TRACEBACKQQ

tune

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

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T

Syntax

Linux and Mac OS: `-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 Intel® EM64T systems, only *keyword* `pn4` is valid.

Alternate Options

None

u (Linux* and Mac OS*)

See [warn.](#)

u (Windows*)

Undefines all previously defined preprocessor values.

IDE Equivalent

Windows: **Preprocessor > Undefine All Preprocessor Definitions**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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 `/undefine` option.

Alternate Options

None

See Also

`undefine` compiler option

U

See [undefine](#).

undefine

Undefines any definition currently in effect for the specified symbol.

IDE Equivalent

Windows: **Preprocessor > Undefine Preprocessor Definitions**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: None

Windows: `/undefine: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.

To undefine all previously defined preprocessor values, use the `/u` option.

Alternate Options

Linux and Mac OS: `-Uname`

Windows: `/Uname`

See Also

`u` (Windows) compiler option

unroll, Qunroll

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

IDE Equivalent

Windows: **Optimization > Loop Unroll Count**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 Itanium®-based processors, you can only specify a value of 0.

Default

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

`/Qunroll`

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: `-funroll-loops`

Windows: `/unroll:n`

See Also

Optimizing Applications: Loop Unrolling

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, Intel® EM64T, Intel® Itanium® architecture
/Quse-asm: Intel® Itanium® architecture

Syntax

Linux and Mac OS: -use-asm
 -no-use-asm

Windows: /Quse-asm
 /Quse-asm-

Arguments

None

Default

OFF The compiler produces objects directly.

Description

This option tells the compiler to produce objects through the assembler.

Alternate Options

Linux: -use_asm
Mac OS: None
Windows: /Quse_asm

v

Specifies that driver tool commands should be displayed and executed.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-v`

Windows: None

Arguments

None

Default

OFF No tool commands are shown.

Description

This option specifies that driver tool commands should be displayed and executed.

If you want to display processing information (pass information and source file names), specify option `watch:all`.

Alternate Options

Linux and Mac OS: `-watch cmd`

Windows: `/watch:cmd`

See Also

`dryrun` compiler option

`watch` compiler option

V (Linux* and Mac OS*)

See [logo](#).

V (Windows*)

See [bintext](#).

vec-report, Qvec-report

Controls the diagnostic information reported by the vectorizer.

IDE Equivalent

Windows: **Diagnostics > Vectorizer Diagnostic Level**

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T

Syntax

Linux and Mac OS: `-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

OFF The compiler does not generate a vectorization report.

Description

This option controls the diagnostic information reported by the vectorizer.

If you do not specify *n*, it is the same as specifying `-vec-report1` (Linux and Mac OS) or `/Qvec-report1` (Windows).

If this option is specified on the command line, the report is sent to stdout.

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

Linux: `-vec_report`

Mac OS: None

Windows: `/Qvec_report`

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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 `FORTTRAN` 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: 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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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 or /warn:none)

Diagnostics > Treat Warnings as Errors (/warn: [no]errors)

Diagnostics > Treat Fortran Standard Warnings as Errors (/warn: [no]stderrs)

Diagnostics > Compile-Time Diagnostics (/warn:all or /warn:none)

Diagnostics > Warn for Undeclared Symbols (/warn: [no]declarations)

Diagnostics > Warn for Unused Variables (/warn: [no]unused)

Diagnostics > Warn When Removing %LOC (/warn: [no]ignore_loc)

Diagnostics > Warn When Truncating Source Line (/warn: [no]truncated_source)

Diagnostics > Warn for Unaligned Data (/warn: [no]alignments)

Diagnostics > Warn for Uncalled Routine (/warn: [no]uncalled)

Diagnostics > Suppress Usage Messages (/warn: [no]usage)

Diagnostics > Check Routine Interface (/warn: [no]interfaces)

Linux: None

Mac OS: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -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.

<code>[no] interfaces</code>	Determines whether the compiler checks the interfaces of all SUBROUTINEs called and FUNCTIONs invoked in your compilation against an external set of interface blocks.
<code>[no] stderrs</code>	Determines whether warnings about Fortran standard violations are changed to errors.
<code>[no] truncated_source</code>	Determines whether warnings occur when source exceeds the maximum column width in fixed-format files.
<code>[no] uncalled</code>	Determines whether warnings occur when a statement function is never called
<code>[no] unused</code>	Determines whether warnings occur for declared variables that are never used.
<code>[no] usage</code>	Determines whether warnings occur for questionable programming practices.
<code>all</code>	Enables all warning messages.

Default

<code>alignments</code>	Warnings are issued about data that is not naturally aligned.
<code>general</code>	All information-level and warning-level messages are enabled.
<code>usage</code>	Warnings are issued for questionable programming practices.
<code>nodeclarations</code>	No errors are issued for undeclared symbols.
<code>noerrors</code>	Warning-level messages are not changed to error-level messages.
<code>noignore_loc</code>	No warnings are issued when <code>%LOC</code> is stripped from an argument.
<code>nointerfaces</code>	The compiler does not check interfaces of SUBROUTINEs called and FUNCTIONs invoked in your compilation against an external set of interface blocks.
<code>nostderrors</code>	Warning-level messages about Fortran standards violations are not changed to error-level messages.
<code>notruncated_source</code>	No warnings are issued when source exceeds the maximum column width in fixed-format files.
<code>nouncalled</code>	No warnings are issued when a statement function is not called.
<code>nounused</code>	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
<code>warn none</code>	Disables all warning messages. This is the same as specifying <code>nowarn</code> .
<code>warn noalignments</code>	Disables warnings about data that is not naturally aligned.
<code>warn declarations</code>	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.
<code>warn errors</code>	Tells the compiler to change all warning-level messages to error-level messages; this includes warnings about Fortran standards violations.
<code>warn nogeneral</code>	Disables all informational-level and warning-level diagnostic messages.
<code>warn ignore_loc</code>	Enables warnings when %LOC is stripped from an actual argument.
<code>warn interfaces</code>	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 <code>gen-interfaces</code> 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 <code>include (-I)</code> or <code>-module</code> option.
<code>warn stderrors</code>	Tells the compiler to change all warning-level messages about Fortran standards violations to error-level messages. This option sets the <code>std95</code> option (Fortran 95 standard). If you want Fortran 90 standards violations to become errors, you should specify <code>warn stderrors</code> and <code>std90</code> .
<code>warn truncated_source</code>	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 <code>extend_source</code> option. The <code>warn truncated_source</code> 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.
<code>warn uncalled</code>	Enables warnings when a statement function is never called.
<code>warn unused</code>	Enables warnings for variables that are declared but never used.
<code>warn nousage</code>	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 stderrs`. 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 stderrs`.

On Windows systems: In the Property Pages, **Custom** means that diagnostics will be specified on an individual basis.

Alternate Options

<code>warn none</code>	Linux and Mac OS: <code>-nowarn, -w, -W0, -warn nogeneral</code> Windows: <code>/nowarn, /w, /W0, /warn:nogeneral</code>
<code>warn declarations</code>	Linux and Mac OS: <code>-implicitnone, -u</code> Windows: <code>/4Yd</code>
<code>warn nodeclarations</code>	Linux and Mac OS: <code>None</code> Windows: <code>/4Nd</code>
<code>warn general</code>	Linux and Mac OS: <code>-W1</code> Windows: <code>/W1</code>
<code>warn nogeneral</code>	Linux and Mac OS: <code>-W0, -w, -nowarn, -warn none</code> Windows: <code>/W0, /w, /nowarn, /warn:none</code>
<code>warn stderrs</code>	Linux and Mac OS: <code>-e90, -e95</code> Windows: <code>/4Ys</code>
<code>warn nostderrors</code>	Linux and Mac OS: <code>None</code> Windows: <code>/4Ns</code>
<code>warn nousage</code>	Linux and Mac OS: <code>-cm</code> Windows: <code>/cm</code>
<code>warn all</code>	Linux and Mac OS: <code>-warn</code> Windows: <code>/warn</code>

watch

Tells the compiler to display certain information to the console output window.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-watch [keyword]`
`-nowatch`

Windows: `/watch[:keyword]`
`/nowatch`

Arguments

keyword Determines what information is displayed. Possible values are:

<code>none</code>	Disables <code>cmd</code> and <code>source</code> .
<code>[no] cmd</code>	Determines whether driver tool commands are displayed and executed.
<code>[no] source</code>	Determines whether the name of the file being compiled is displayed.
<code>all</code>	Enables <code>cmd</code> and <code>source</code> .

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
<code>watch</code> <code>none</code>	Tells the compiler to not display pass information and source file names to the console output window. This is the same as specifying <code>nowatch</code> .
<code>watch</code> <code>cmd</code>	Tells the compiler to display and execute driver tool commands.
<code>watch</code>	Tells the compiler to display the name of the file being compiled.

`source`

`watch` 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: `-v`
Windows: None

See Also

`v` compiler option

WB

Turns a compile-time bounds check into a warning.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: -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 the version strings of the Fortran driver and the compiler.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-what`

Windows: `/what`

Arguments

None

Default

OFF The version strings are not displayed.

Description

This option tells the compiler to display the version strings of the Fortran driver and the compiler.

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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: 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: None

Windows: /MG

WI

Passes options to the linker for processing.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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

Directs the compiler to generate specialized and optimized code for the processor that executes your program.

IDE Equivalent

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

Linux: None

Architectures

IA-32, Intel® EM64T

Syntax

Linux and Mac OS: `-xprocessor`

Windows: `/Qxprocessor`

Arguments

processor Is the processor for which you want to target your program. Possible values are:

- K** Code is optimized for Intel® Pentium® III and compatible Intel processors.
- W** Code is optimized for Intel Pentium 4 and compatible Intel processors.
- N** Code is optimized for Intel Pentium 4 and compatible Intel processors with Streaming SIMD Extensions 2. The resulting code may contain unconditional use of features that are not supported on other processors. This option also enables new optimizations in addition to Intel processor-specific optimizations including advanced data layout and code restructuring optimizations to improve memory accesses for Intel processors.
- B** Code is optimized for Intel Pentium M and compatible Intel processors. This option also enables new optimizations in addition to Intel processor-specific optimizations.
- P** Code is optimized for Intel® Core™ Duo processors, Intel® Core™ Solo processors, Intel® Pentium® 4 processors with Streaming SIMD Extensions 3, and compatible Intel processors with Streaming SIMD Extensions 3. The resulting code may contain unconditional use of features that are not supported on other processors. This option also enables new optimizations in addition to Intel processor-specific optimizations including advanced data layout and code restructuring optimizations to improve memory accesses for Intel processors.
- T** Code is optimized for Intel® Core™2 Duo processors, Intel® Core™2

Extreme processors, and the Dual-Core Intel® Xeon® processor 5100 series.

This option also enables new optimizations in addition to Intel processor-specific optimizations including advanced data layout and code restructuring optimizations to improve memory accesses for Intel processors.

Default

Windows and Linux IA-32 systems: OFF	On Windows and Linux IA-32 systems, no processor-specific code is generated by the compiler. On Intel® EM64T systems, code is optimized for Intel Pentium 4 and compatible processors. On Mac OS systems, code is optimized for Intel® Core™ Duo processors and Intel® Pentium® 4 processors with Streaming SIMD Extensions 3 (SSE3) instruction support.
Windows and Linux Intel® EM64T systems: -xW	
Mac OS systems: -xP	

Description

This option directs the compiler to generate specialized and optimized code for the Intel® processor that executes your program. It lets you target your program to run on a specific Intel processor.

The resulting code may contain unconditional use of features that are not supported on other processors.

On Intel® EM64T systems, *W*, *P*, and *T* are the only valid *processor* values.

On Mac OS systems, *P* is the only valid *processor* value. On these systems, it is the default and is always set.

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: *T*, *P*, *B*, *N*, *W*, *K*.

Do not use these options if you are executing a program on a processor that is not an Intel® processor. If you use these options on a non-compatible processor to compile the main program (in Fortran) or the function `main()` in C/C++, the program may fail with an illegal instruction exception or display other unexpected behavior.

In particular, such programs compiled with *processor* values *N*, *B*, or *P* will display a fatal run-time error if they are executed on unsupported processors. For more information, see your Optimizing Applications guide.

Alternate Options

None

See Also

ax compiler option

X

Removes standard directories from the include file search path.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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.

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.

Alternate Options

Linux and Mac OS: `-nostdinc`

Windows: `/noinclude`

See Also

`I` compiler option

Xlinker

Passes a linker option directly to the linker.

IDE Equivalent

None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-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. See 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: None

Architectures

IA-32, Intel® EM64T, Intel® Itanium® architecture

Syntax

Linux and Mac OS: `-zero`
`-nozero`

Windows: `/Qzero`
`/Qzero-`

Arguments

None

Default

OFF 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) 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 Reference of Compiler Options

This section provides cross-reference tables of compiler options used on Windows* operating systems and on Linux* and Mac OS* 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 IA-32-based systems.
i32em	The option is available on IA-32-based systems with Intel® Extended Memory 64 Technology (Intel® EM64T).
i64	The option is available on Itanium®-based systems.

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 Options

The following cross-reference table shows all supported Windows options and equivalent Linux and Mac OS options, if any. If an equivalent option in the Linux and Mac OS Option column is restricted to Linux systems, it is labeled "Linux only".

Windows Option	Linux and Mac OS 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	/4L72 -72

			72, 80, or 132; same as the <code>/extend_source</code> option.	
<code>/4Na, /4Ya</code>	None		Determines where local variables are stored. <code>/4Na</code> is the same as <code>/Qsave</code> . <code>/4Ya</code> is the same as <code>/Qautomatic</code> .	ON
<code>/4Naltparam, /4Yaltparam</code>	None		Determines whether alternate syntax is allowed for PARAMETER statements; same as the <code>/altparam</code> option).	ON
<code>/4Nb, /4Yb</code>	None		Determines whether checking is performed for run-time failures (same as the <code>/check</code> option).	OFF
<code>/4Nd, /4Yd</code>	-implicitnone or -u (for <code>/4Yd</code>)		Determines whether error messages are issued for undeclared symbols. <code>/4Nd</code> is the same as <code>/warn:nodeclarations</code> . <code>/4Yd</code> is the same as <code>/warn:declarations</code> .	OFF
<code>/4Nf, /4Yf</code>	None		Specifies the format for source files. <code>/4Nf</code> is the same as <code>/fixed</code> . <code>/4Yf</code> is the same as <code>/free</code> .	OFF
<code>/4Ns, /4Ys</code>	-e95 or -e90 (for <code>/4Ys</code>)		Determines whether the compiler changes warning messages about Fortran standards violations to error messages. <code>/4Ns</code> is the same as <code>/warn:nostderrors</code> . <code>/4Ys</code> is the same as <code>/warn:stderrors</code> .	OFF
<code>/4R8, /4R16</code>	None		Specifies the default KIND for real and complex variables; same as the <code>/real_size</code> option.	OFF
<code>/4Yportlib</code>	None		Links against the library of portability routines.	ON
<code>/align[:keyword]</code>	-align [keyword]		Tells the compiler how to	keywords:

		align certain data items.	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.	ON
/architecture:keyword (i32, i32em)	-arch keyword (i32, i32em)	Determines the version of the architecture for which the compiler generates instructions.	keyword: pn4
/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: protect_ constants source_ include nobsc nobuffered_io nobyterecl nocc_omp nodummy_ aliases nominus0 nounderscore no2underscores nowriteable-strings
/auto	-auto	Causes all variables to be allocated to the run-time stack; same as the /automatic option.	OFF
/automatic	-automatic	Causes all variables to be allocated to the run-time stack; same as the /auto option.	OFF
/bintext	None	Places the text string specified into the object file (.obj) being generated	OFF

		by the compiler.	
/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.	OFF
/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
/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.	OFF
/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	Specifies the type of	keywords:

Intel(R) Fortran Compiler Options

	OS option takes different keywords	debugging information generated by the compiler in the object file.	full (IDE) minimal (command line)
/debug-parameters[:keyword]	-debug-parameters [keyword]	Tells the compiler to generate debug information for PARAMETERS used in a program.	keyword: none
/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 to send output to stdout.	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
/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	OFF

		to be recognized as a file to be preprocessed by the Fortran preprocessor.	
/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 program or dynamic- link library; same as the /exe option.	OFF
/FI	-FI	Specifies source files are in fixed format; same as the /fixed option.	
/fixed	-fixed	Specifies source files are in fixed format.	
/fltconsistency	-fltconsistency	Enables improved floating-point consistency.	OFF
/Fm[file]	None	Tells the linker to generate a link map file;	OFF

Intel(R) Fortran Compiler Options

		same as the <code>/map</code> option.
<code>/Fo<file></code>	None	Specifies the name for an OFF object file; same as the <code>/object</code> option.
<code>/fp:<keyword></code>	<code>-fp-model <keyword></code>	Controls the semantics of <code>/fp:fast</code> floating-point calculations.
<code>/fpconstant</code>	<code>-fpconstant</code>	Tells the compiler that single-precision constants assigned to double-precision variables should be evaluated in double precision. OFF
<code>/fpe:<n></code>	<code>-fpe<n></code>	Specifies floating-point exception handling for the main program at run-time. n: 3
<code>/fpp</code>	<code>-fpp</code>	Runs the Fortran preprocessor on source files before compilation. OFF
<code>/fpscomp[:keyword]</code>	<code>-fpscomp [keyword]</code>	Specifies compatibility with Microsoft* Fortran PowerStation or Intel® Fortran. keyword: libs
<code>/FR</code>	<code>-FR</code>	Specifies source files are determined by file suffix in free format; same as the <code>/free</code> option.
<code>/free</code>	<code>-free</code>	Specifies source files are determined by file suffix in free format.
<code>/G{1 2}</code> (i64 only)	<code>-tpp{1 2}</code> (i64 only; Linux only)	Optimizes application performance for Intel® Itanium® processors. /G2 -tpp2
<code>/G2-p9000</code> (i64 only)	<code>-mtune itanium2-p9000</code> (i64 only; Linux only)	Optimizes for Dual-Core Intel® Itanium® 2 Processor 9000 Sequence processors. OFF
<code>/G{5 6 7}</code> (i32, i32em)	<code>-tpp{5 6 7}</code> (i32, i32em; Linux only)	Optimizes application performance for IA-32 and Intel® EM64T processors. /G7 -tpp7
<code>/GB</code>	None	Optimizes for Intel® Pentium® Pro, Pentium® II and Pentium® III processors; same as the <code>/G6</code> option. OFF
<code>/Ge</code>	None	Enables stack-checking for all functions. OFF

/gen-interfaces	-gen-interfaces	Tells the compiler to generate an interface block for each routine in a source file.	OFF
/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
/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>	-I<dir>	Specifies a directory to add to the include path; same as the /I option.	OFF
/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

Intel(R) Fortran Compiler Options

/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: ON Linux: OFF
/map[:file]	None	Tells the linker to generate a link map file.	OFF
/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 i32em: OFF
/MLd	None	Specifies a single-threaded, static, debug library; same as specifying options /libs:static	i32, i64: ON i32em: OFF

/module:<path>	-module <path>	/dbglibs. 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 i32, i64: OFF for unresolved references i32em: ON in a multithreaded, static run-time library.	
/MTd	None	Tells the linker to search i32, i64: OFF for unresolved references i32em: ON in a multithreaded, static, debug run-time library.	
/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: uppercase 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
/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
/nodefine	-nodefine	Specifies that all preprocessor definitions apply only to fpp and not to Intel® Fortran conditional compilation directives.	OFF
/noinclude	-X	Removes standard directories from the include file search path;	OFF

Intel(R) Fortran Compiler Options

<code>/nowarn</code>	<code>-nowarn</code>	same as the <code>/x</code> option. Suppresses all warning messages.	OFF
<code>/O1</code>	<code>-O1</code>	Enables optimizations for speed and disables some optimizations that increase code size and affect speed.	OFF
<code>/O2</code>	<code>-O2</code>	Enables optimizations for speed. This is the generally recommended optimization level.	ON
<code>/O3</code>	<code>-O3</code>	Enables <code>/O2</code> optimizations plus more aggressive optimizations.	OFF
<code>/Ob<n></code>	<code>-Ob<n></code>	Specifies the level of inline function expansion. <code>/Ob2</code> if <code>/O2</code> is in effect <code>/Ob0</code> if <code>/Od</code> is specified <code>n = 0, 1, or 2.</code>	
<code>/object:<file></code>	None	Specifies the name for an object file.	OFF
<code>/Od</code>	<code>-O0</code>	Disables optimizations.	OFF
<code>/Og</code>	None	Enables global optimizations.	ON
<code>/Op</code>	<code>-mp</code>	Enables improved floating-point consistency.	OFF
<code>/optimize:<n></code>	<code>-O<n></code>	Affects optimizations performed by the compiler; <code>n = 1, 2, 3, or 4.</code>	OFF
<code>/Os</code>	None	Enables most speed optimizations, but disables some optimizations that increase code size for a small speed benefit.	ON
<code>/Ot</code>	None	Enables all speed optimizations.	ON
<code>/Ox</code>	<code>-O2</code>	Same as the <code>/O2</code> option.	ON
<code>/Oy-</code> (i32 only)	<code>-fp</code> (i32, i32em)	Disallows use of EBP as a general-purpose register in optimizations.	OFF
<code>/pad-source</code>	<code>-pad-source</code>	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.	<code>/nopad_source</code>
<code>/pdbfile[:file]</code>	None	Specifies that any debug	<code>/nopdbfile</code>

		information generated by the compiler should be saved to a program database file.	
/preprocess_only	-preprocess_only	Causes the Fortran preprocessor to send output to a file, which is named by default. Requires option -fpp.	OFF
/Qansi-alias	-ansi-alias	Tells the compiler to assume the program adheres to the Fortran 95 Standard type aliasability rules.	ON
/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.	ON
/Qautodouble	-autodouble	Makes default real and complex variables 8 bytes long; same as the /real_size:64 option.	OFF
/Qax<p> (i32, i32em)	-ax<p> (i32, i32em)	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.	ON
/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.	OFF

Intel(R) Fortran Compiler Options

/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
/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
/Qfnsplit (i32, i64)	-fnsplit (i64 only; Linux only)	Enables function splitting.	OFF
/Qfp_port (i32 only)	-fp-port (i32, i32em)	Rounds floating-point results after floating-point operations, so rounding to user-declared precision happens at assignments and type conversions (some impact on speed).	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.	OFF
/Qfpstkchk (i32, i32em)	-fpstkchk (i32, i32em)	Generates extra code after every function call to ensure that the FP (floating-point) stack is in the expected state.	OFF
/Qftz	-ftz	Flushes denormal results to zero.	OFF
/Qglobal-hoist	-global-hoist	Enables certain optimizations that can move memory loads to a point earlier in the	OFF

		program execution than where they appear in the source.	
/QIA64-fr32 (i64 only)	None	Disables use of high floating-point registers.	OFF
/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-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
/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	-ip-no-pinlining	Disables partial inlining.	OFF

Intel(R) Fortran Compiler Options

(i32, i32em) /QIPF-flt-eval- method0 (i64 only)	(i32, i32em) -IPF-flt-eval- method0 (i64 only; Linux 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
/QIPF-fltacc (i64 only)	-IPF-fltacc (i64 only; Linux only)	Tells the compiler to apply optimizations that affect floating-point accuracy.	OFF
/QIPF-fma (i64 only)	-IPF-fma (i64 only; Linux only)	Enables the combining of floating-point multiplies and add/subtract operations.	ON
/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.	OFF
/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-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
/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 to lowercase; same as the /names:lowercase	Windows: OFF Linux: ON

		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-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-bandwidth0 for serial compilation; /Qopt-mem-bandwidth1 for parallel compilation
/Qopt-report	-opt-report	Tells the compiler to generate an optimization report to <code>stderr</code> .	OFF
/Qopt-report-file<file>	-opt-report-file<file>	Tells the compiler to generate an optimization report named <code>file</code> .	OFF
/Qopt-report-help	-opt-report-help	Displays the logical names of optimizers available for report generation (using <code>/Qopt_report_phase</code>).	OFF
/Qopt-report-level<level>	-opt-report-level<level>	Specifies the detail level of the optimization report.	level: min
/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	OFF

		containing the specified string.	
/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.	OFF
/Qpad-source	-pad-source	This is the same as specifying option /pad-source.	OFF
/Qpar-report [n]	-par-report [n]	Controls the auto-parallelizer's level of diagnostic messages.	n=1
/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, i32em)	-pc<n> (i32, i32em)	Enables control of floating-point significand precision.	n = 64
/Qprec	-mp1	Improves floating-point precision; disables fewer optimizations and has less impact on performance than /fltconsistency.	OFF
/Qprec-div (i32, i32em)	-prec-div (i32, i32em)	Disables floating point division-to-multiplication optimization resulting in more accurate division results; some speed impact.	OFF
/Qprec-sqrt (i32, i32em)	-prec-sqrt (i32, i32em)	Improves precision of square root implementations.	OFF
/Qprefetch (i64 only)	-prefetch (i64 only; Linux only)	Enables prefetch insertion optimization (requires -O3).	ON
/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	OFF

		the profiling summary file.	
/Qprof-format-32 (i32, i64)	-prof-format-32 (i32, i64)	Produces profile data with 32-bit counters.	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, i32em)	-rcd (i32, i32em)	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.	OFF
/Qscalar-rep (i32 only)	-scalar-rep (i32 only)	Enables scalar replacement performed during loop transformation (requires /O3).	OFF
/Qsfa1ign[n] (i32 only)	None	Specifies stack alignment for functions. n is 8 or 16.	
/Qsox	-sox	Tells the compiler to save the compiler options and version number in the executable.	OFF
/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
/Qtrapuv	-fttrapuv	Initializes local variables	OFF

Intel(R) Fortran Compiler Options

<code>/Qunroll[:n]</code>	<code>-unroll[n]</code>	to Not a Number (NaN). Tells the compiler the maximum number of times to unroll loops; same as the <code>/unroll[:n]</code> option.	OFF
<code>/Quppercase</code>	<code>-uppercase</code>	Causes the compiler to ignore case differences in identifiers and to convert external names to uppercase; same as the <code>/names:uppercase</code> option.	Windows: ON Linux: OFF
<code>/Quse-asm</code> (i32 only)	<code>-use-asm</code>	Tells the compiler to produce objects through the assembler.	OFF
<code>/Quse_vcdebug</code> (i32 only)	None	Tells the compiler to issue debug information compatible with the Visual C++ debugger.	OFF
<code>/Qvc6</code> (i32 only)	None	Specifies compatibility with Visual C++ 6.0.	OFF
<code>/Qvc7</code> (i32 only)	None	Specifies compatibility with Microsoft* Visual Studio .NET 2002.	OFF
<code>/Qvc7.1</code> (i32, i32em)	None	Specifies compatibility with Microsoft* Visual Studio .NET 2003.	OFF
<code>/Qvc8</code> (i32, i32em)	None	Specifies compatibility with Microsoft* Visual Studio .NET 2005.	OFF
<code>/Qvec-report[n]</code> (i32, i32em)	<code>-vec-report[n]</code> (i32, i32em)	Controls the diagnostic information reported by the vectorizer.	$n = 1$
<code>/Qvms</code>	<code>-vms</code>	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 <code>/vms</code> option.	OFF
<code>/Qx<p></code> (i32, i32em)	<code>-x<p></code> (i32, i32em)	Generates the minimum set of processor-specific instructions required for the processor that executes your program. The <code>p</code> indicates the processor type.	i32: OFF i32em: <code>/QxW</code>

/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.	OFF
/reentrancy:<keyword>	-reentrancy <keyword>	Tells the compiler to generate reentrant code to support a multithreaded application.	OFF
/RTCu	None	Causes checking to occur for uninitialized variables; same as the /check:uninit option.	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 nonstandard language elements.	OFF
/static	-static (Linux only)	Prevents linking with shared libraries.	ON
/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: OFF i32em: ON
/traceback	-traceback	Tells the compiler to generate extra information in the object file to provide source file	OFF

		traceback information when a severe error occurs at run time.	
/tune:<keyword> (i32, i32em)	-tune <keyword> (i32, i32em)	Determines the version of the architecture for which the compiler generates instructions.	keyword: pn4
/u	None Note: the Linux and Mac OS 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.	OFF
/us	-us	Tells the compiler to append an underscore character to external user-defined names; same as the /assume:underscore option.	OFF
/Vstring	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 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
/warn:<keyword>	-warn <keyword>	Specifies diagnostic	keywords: alignments

Compiler Options

		messages to be issued by the compiler.	general usage nodeclarations noerrors noignore_loc nointerfaces nostderrors notruncated_source nouncalled nounused nowatch
/watch[:keyword]	-watch [keyword]	Tells the compiler to display certain information to the console output window.	
/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 components of derived types on the smaller of the size boundary specified or the boundary that will naturally align them;	n = 16

		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 Options to Windows Options

The following cross-reference table shows all supported Linux and Mac OS options and equivalent Windows options, if any. If an option in the Linux and Mac OS Option column is restricted to Linux systems, it is labeled "Linux only".

Linux and Mac OS 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	ON

		parentheses) for PARAMETER 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, i32em)	/architecture:<keyword> (i32, i32em)	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: protect_ constants source_ include_ nobsc nobuffered_io nobyterecl nocc_omp nodummy_ aliases nominus0 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 the	OFF

		-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, i32em)	/Qax<p> (i32, i32em)	Generates processor-specific code if there is a performance benefit. The p indicates the processor type.	Linux: OFF Mac OS: -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 OFF conditions at run time.
-cm	/cm	Disables all OFF messages about questionable programming practices; same as specifying option -warn nousage.
-common-args	/Qcommon-args	Tells the compiler OFF that dummy (formal) arguments to procedures share memory locations with other dummy arguments or with COMMON variables that are assigned.
-complex-limited-range	/Qcomplex-limited-range	Enables the use OFF of basic algebraic expansions of some arithmetic operations involving data of type COMPLEX.
-convert <keyword>	/convert:<keyword>	Specifies the keyword: format of native unformatted files containing numeric data.
-cpp	/Qcpp	Runs the Fortran OFF preprocessor on source files prior to compilation.
-cxxlib-<mode>	None	Tells the compiler OFF to link using certain C++ run- time libraries.
-D<name> [=value]	/D<name> [=value]	Defines a symbol OFF name that can be associated with

Intel(R) Fortran Compiler Options

		an optional value.	
<code>-d_lines</code>	<code>/d_lines</code>	Compiles debugging statements indicated by the letter D in column 1 of the source code.	OFF
<code>-DD</code>	<code>/Qd_lines</code>	Compiles debugging statements indicated by the letter D in column 1 of the source code; same as the <code>-d_lines</code> option.	OFF
<code>-debug <keyword></code>	<code>/debug:<keyword></code> Note: the Windows option takes different keywords	Specifies settings that enhance debugging.	OFF
<code>-debug-parameters [keyword]</code>	<code>/debug-parameters[:keyword]</code>	Tells the compiler to generate debug information for PARAMETERS used in a program.	keyword: none
<code>-double_size <size></code>	<code>/double_size:<size></code>	Defines the default KIND for DOUBLE PRECISION and DOUBLE COMPLEX variables.	size = 64
<code>-dps</code>	<code>/Qdps</code>	Specifies that the alternate syntax for PARAMETER statements is allowed; same as the <code>-altparam</code> option.	ON
<code>-dryrun</code>	None	Specifies that driver tool commands should be shown but not executed.	OFF
<code>-dynamic-linker<file></code> (Linux only)	None	Specifies a dynamic linker in	OFF

		file other than the default.	
-dynamiclib (i32 only; Mac OS 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
-e95, -e90	/4Ys	Causes the compiler to issue errors instead of warnings for nonstandard Fortran 95 or Fortran 90; 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
-F (Linux only)	None	Causes the Fortran preprocessor to	OFF

Intel(R) Fortran Compiler Options

		send output to a file, which is named by default (same as the <code>-preprocess_only</code> or <code>-P</code> option).	
<code>-f66</code>	<code>/f66</code>	Tells the compiler to use FORTRAN 66 semantics.	OFF
<code>-f77rtl</code>	<code>/f77rtl</code>	Tells the compiler to use FORTRAN 77 run-time behavior.	OFF
<code>-fast</code>	<code>/fast</code>	Maximizes speed across the entire program.	OFF
<code>-fcode-asm</code>	<code>/FAC</code>	Produces an assembly file with optional machine code annotations.	OFF
<code>-FI</code>	<code>/FI</code>	Specifies source files are in fixed format; same as the <code>-fixed</code> option.	determined by file suffix
<code>-finline-functions</code>	None	Enables certain interprocedural optimizations for single file compilation.	ON
<code>-finline-limit=<n></code>	None	Lets you specify the maximum size of a function to be inlined.	OFF
<code>-fixed</code>	<code>/fixed</code>	Specifies source files are in fixed format.	determined by file suffix
<code>-fltconsistency</code>	<code>/fltconsistency</code>	Enables improved floating-point consistency.	OFF
<code>-fmath-errno</code>	None	Tells the compiler that <code>errno</code> can be reliably tested after calls to standard math library functions.	OFF
<code>-fminshared</code>	None	Tells the compiler to treat a compilation unit	OFF

		as a component of a main program and not to link it as a shareable object.	
<code>-fno-alias</code>	None	Specifies that aliasing should not be assumed in the program.	<code>-falias</code>
<code>-fno-fnalias</code>	None	Specifies that aliasing should not be assumed within functions, but should be assumed across calls.	<code>-ffnalias</code>
<code>-fnsplit</code> (i64 only; Linux only)	<code>/Qfnsplit</code> (i32, i64)	Enables function splitting.	OFF
<code>-fno-omit-frame-pointer</code> (i32, i32em)	None	Disables using EBP as a general purpose register so it can be used as a stack frame pointer. This is the same as specifying option <code>-fp</code> .	OFF
<code>-fp</code> (i32, i32em)	<code>/Oy-</code> (i32 only)	Disables using EBP as a general purpose register so it can be used as a stack frame pointer.	OFF
<code>-fp-model <keyword></code>	<code>/fp:<keyword></code>	Controls the semantics of floating-point calculations.	<code>-fp-model fast</code>
<code>-fp-port</code> (i32, i32em)	<code>/Qfp-port</code> (i32 only)	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
<code>-fpconstant</code>	<code>/fpconstant</code>	Tells the compiler that single-	OFF

		precision constants assigned to double-precision variables should be evaluated in double precision.	
-fpe<n>	/fpe:<n>	Specifies floating-point exception handling at run time for the main program.	-fpe3
-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, i32em)	/Qfpstkchk (i32, i32em)	Generates extra code after every function call to ensure that the FP (floating-point) stack is in the expected state.	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 local variables to Not a Number (NaN).	OFF

-ftz	/Qftz	Flushes denormal results to zero.	OFF
-funroll-loops	None	Tells the compiler to unroll user loops based on the default optimization heuristics; same as option -unroll.	OFF
-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
-gen-interfaces	/gen-interfaces	Tells the compiler to generate an interface block for each routine in a source file.	OFF
-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.	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

Intel(R) Fortran Compiler Options

<code>-i-static</code>	None	Links Intel-provided libraries statically.	OFF
<code>-i{2 4 8}</code>	<code>/4I{2 4 8}</code>	Specifies the default KIND for integer and logical variables; same as the <code>-integer_size</code> option.	<code>-i4</code>
<code>-implicitnone</code>	<code>/4Yd</code>	Sets the default type of a variable to undefined.	OFF
<code>-inline-debug-info</code>	<code>/Qinline-debug-info</code>	Produces enhanced source position information for inlined code.	OFF
<code>-inline-factor=<n></code>	<code>/Qinline-factor=<n></code>	Specifies the percentage multiplier that should be applied to all inlining options that define upper limits.	OFF
<code>-inline-forceinline</code>	<code>/Qinline-forceinline</code>	Specifies that an inline routine should be inlined whenever the compiler can do so.	OFF
<code>-inline-max-per-compile=<n></code>	<code>/Qinline-max-per-compile=<n></code>	Specifies the maximum number of times inlining may be applied to an entire compilation unit.	OFF
<code>-inline-max-per-routine=<n></code>	<code>/Qinline-max-per-routine=<n></code>	Specifies the maximum number of times the inliner may inline into a particular routine.	OFF
<code>-inline-max-size=<n></code>	<code>/Qinline-max-size=<n></code>	Specifies the lower limit for the size of what the inliner considers to be a large routine.	OFF
<code>-inline-max-total-size=<n></code>	<code>/Qinline-max-total-size=<n></code>	Specifies how much larger a	OFF

		routine can normally grow when inline expansion is performed.	
<code>-inline-min-size=<n></code>	<code>/Qinline-min-size=<n></code>	Specifies the upper limit for the size of what the inliner considers to be a small routine.	OFF
<code>-intconstant</code>	<code>/intconstant</code>	Tells the compiler to use Fortran 77 semantics to determine the KIND for integer constants.	OFF
<code>-integer_size <size></code>	<code>/integer_size:<size></code>	Specifies the default KIND for integer and logical variables.	size = 32
<code>-ip</code>	<code>/Qip</code>	Enables additional single-file interprocedural optimizations.	OFF
<code>-ip-no-inlining</code>	<code>/Qip-no-inlining</code>	Disables full and partial inlining enabled by <code>-ip</code> .	OFF
<code>-ip-no-pinlining</code>	<code>/Qip-no-pinlining</code>	Disables partial inlining.	OFF
<code>-IPF-flt-eval-method0</code> (i64 only; Linux only)	<code>/QIPF-flt-eval-method0</code> (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
<code>-IPF-fltacc</code> (i64 only; Linux only)	<code>/QIPF-fltacc</code> (i64 only)	Tells the compiler to apply optimizations that affect floating-point accuracy.	OFF
<code>-IPF-fma</code> (i64 only; Linux only)	<code>/QIPF-fma</code> (i64 only)	Enables the combining of floating-point multiplies and add/subtract	ON

Intel(R) Fortran Compiler Options

-IPF-fp-relaxed (i64 only; Linux only)	/QIPF-fp-relaxed (i64 only)	operations. 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 OFF
-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-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 loop-carried memory dependency in any loop following an IVDEP directive.	OFF
-Kpic, -KPIC (Linux only)	None	This is a deprecated option; use -fpic instead.	OFF
-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
-map-opts	/Qmap-opts	Converts one or more Linux* compiler options to their equivalent on a Windows* system (or vice versa).	OFF
-mcmmodel=mem_model (i32em only; Linux only)	None	Tells the compiler to use a specific memory model to generate code and store data.	-mcmmodel=small
-mieeee-fp	/fltconsistency	Tells the compiler to use IEEE floating point comparisons. This is the 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

Intel(R) Fortran Compiler Options

<code>-module <dir></code>	<code>/module:<path></code>	Specifies the directory where module files should be placed when created and where they should be searched for.	OFF
<code>-mp</code>	<code>/Op</code>	Enables improved floating-point consistency.	OFF
<code>-mp1</code>	<code>/Qprec</code>	Improves floating-point precision; disables fewer optimizations and has less impact on performance than <code>-fltconsistency</code> or <code>-mp</code> .	OFF
<code>-mrelax</code>	None	Tells the compiler to pass linker option <code>-relax</code> to the linker.	OFF
<code>-mtune=<keyword></code>	None	Performs optimizations for a particular CPU.	OFF
<code>-mtune=itanium2-p9000</code> (i64 only; Linux only)	<code>/G2-p9000</code> (i64 only)	Optimizes for Dual-Core Intel® Itanium® 2 Processor 9000 Sequence processors.	OFF
<code>-names <keyword></code>	<code>/names:<keyword></code>	Specifies how source code identifiers and external names are interpreted.	keyword: Linux: lowercase Windows: uppercase
<code>-nbs</code>	<code>/nbs</code>	Tells the compiler to treat the backslash character (\) as a normal character in character literals; same as the <code>-assume nobsc</code> option.	ON
<code>-no-cpprt</code>	None	Tells the compiler to use the default run-time libraries and not link to any	OFF

		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	OFF

		memory loads to a point earlier in the program execution than where they appear in the source.	
-noinclude	/noinclude	Prevents the compiler from searching in a directory previously added to the include path for files specified in an INCLUDE statement.	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-defined names; same as the -assume nounderscore option.	OFF
-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	OFF

		optimizations that increase code size and affect speed.	
-O2	/O2	Enables optimizations for speed. This is the generally recommended optimization level.	ON
-O3	/O3	Enables -O2 optimizations plus more aggressive optimizations.	OFF
-Ob<n>	/Ob<n>	Specifies the level of inline function expansion. <i>n</i> = 0, 1, or 2.	-Ob2 if -O2 is in effect -Ob0 if -O0 is specified
-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-profile (Linux only)	/Qopenmp-profile	Enables analysis of OpenMP* applications.	OFF
-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-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

Intel(R) Fortran Compiler Options

<code>-opt-report</code>	<code>/Qopt-report</code>	Tells the compiler to generate an optimization report to <code>stderr</code> .	OFF
<code>-opt-report-file<file></code>	<code>/Qopt-report-file<file></code>	Tells the compiler to generate an optimization report named <code>file</code> .	OFF
<code>-opt-report-help</code>	<code>/Qopt-report-help</code>	Displays the logical names of optimizers available for report generation using <code>-opt-report-phase</code> .	OFF
<code>-opt-report-level<level></code>	<code>/Qopt-report-level<level></code>	Specifies the detail level of the optimization report.	level: min
<code>-opt-report-phase<phase></code>	<code>/Qopt-report-phase<phase></code>	Specifies the optimizer phase to generate reports for.	OFF
<code>-opt-report-routine[string]</code>	<code>/Qopt-report-routine[string]</code>	Generates a report on all routines or the routines containing the specified <code>string</code> .	OFF
<code>-p</code>	None	Compiles and links for function profiling with <code>gprof(1)</code> .	OFF
<code>-P</code>	<code>/P</code>	Causes the Fortran preprocessor to send output to a file, which is named by default (same as the <code>-preprocess_only</code> or <code>-F</code> option).	OFF
<code>-pad</code>	<code>/Qpad</code>	Enables the changing of the variable and array memory layout.	OFF
<code>-pad-source</code>	<code>/pad-source</code>	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
-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, i32em)	/Qpc<n> (i32, i32em)	Enables control of floating-point significand precision.	n = 80
-pg	None	Compiles and links for function profiling with gprof (1).	OFF
-prec-div (i32, i32em)	/Qprec-div (i32, i32em)	Disables floating point division-to-multiplication optimization resulting in more accurate division results; some speed impact.	OFF
-prec-sqrt (i32, i32em)	/Qprec-sqrt (i32, i32em)	Improves precision of square root implementations.	OFF
-prefetch (i64 only; Linux only)	/Qprefetch (i64 only)	Enables prefetch insertion	ON

Intel(R) Fortran Compiler Options

-preprocess_only	/preprocess_only	optimization (requires -O3). 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-format-32 (i32, i64)	/Qprof-format-32 (i32, i64)	Produces profile data with 32-bit counters.	OFF
-prof-gen	/Qprof-gen	Instruments a program for profiling.	OFF
-prof-gen-sampling	/Qprof-gen-sampling	Prepares application executables for hardware profiling (sampling) and causes the compiler to generate source code mapping information.	OFF
-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

<code>-Qinstall<dir></code>	None	Specifies the root directory where the compiler installation was performed.	OFF
<code>-Qlocation,string,dir</code>	<code>/Qlocation,string,dir</code>	Specifies a directory as the location of the specified tool in string.	OFF
<code>-Qoption,string,options</code>	<code>/Qoption,string,options</code>	Passes options to the specified tool in string.	OFF
<code>-qp</code>	None	Compiles and links for function profiling with <code>prof(1)</code> .	OFF
<code>-r8</code>	<code>/4R8</code>	Defines REAL declarations, constants, functions, and intrinsics as DOUBLE PRECISION (REAL*8), and defines COMPLEX declarations, constants, functions, and intrinsics as DOUBLE COMPLEX (COMPLEX*16).	OFF
<code>-r16</code>	<code>/4R16</code>	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	OFF

Intel(R) Fortran Compiler Options

-rcd (i32, i32em)	/Qrcd (i32, i32em)	COMPLEX*32. 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
-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-libcxa (Linux only)	None	Links the Intel libcxa C++ library dynamically.	OFF
-sox	/Qsox	Tells the compiler to save the compiler options	OFF

		and version in the executable.	
-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 compile-time messages for nonstandard language elements.	OFF
-static (Linux only)	/static	Prevents linking with shared libraries.	ON
-static-libcxa (Linux only)	None	Links the Intel libcxa C++ 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 -std or -stand f95	/stand:f95	Causes the compiler to issue messages for language elements that are not standard in Fortran 95.	OFF
-syntax-only	/syntax-only	Specifies that the source file should be checked only for correct syntax.	OFF
-T <file> (Linux only)	None	Tells the linker to read link commands from the specified file.	OFF
-tcheck (Linux only)	/Qtcheck	Enables analysis of threaded applications.	OFF

Intel(R) Fortran Compiler Options

-Tf <file>	/Tf <file>	Tells the compiler to compile the file as a Fortran source file.	OFF
-threads	/threads	Tells the linker to search for unresolved references in a multithreaded run-time library.	i32, i64: OFF i32em: ON
-tpp{1 2} (i64 only; Linux only)	/G{1 2} (i64 only)	Optimizes application performance for Intel® Itanium® processors.	-tpp2 /G2
-tpp{5 6 7} (i32, i32em; Linux only)	/G{5 6 7} (i32, i32em)	Optimizes application performance for IA-32 and Intel® EM64T processors.	-tpp7 /G7
-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, i32em)	/tune:<keyword> (i32, i32em)	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 symbols; same as the -warn declarations option.	OFF
-U<name>	/U<name>	Undefines any definition currently in effect for the specified symbol.	OFF
-unroll [n]	/unroll[:n]	Tells the compiler	OFF

		the maximum number of times to unroll loops.	
-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
-vec-report [n] (i32, i32em)	/Qvec-report [n] (i32, i32em)	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	OFF

		warning messages; same as specifying option -warn none or -warn nogeneral.	
-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 output window.	nowatch
-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
-Wl,o1[,o2,...]	None	Passes options (o1, o2, and so forth) to the linker for processing.	OFF
-Wp,o1[,o2,...]	None	Passes options (o1, o2, and so forth) to the preprocessor.	OFF
-X	/X	Removes standard	OFF

		directories from the include file search path.	
-x<p> (i32, i32em)	/Qx<p> (i32, i32em)	Generates the minimum set of processor-specific instructions required for the processor that executes your program. The p indicates the processor type.	Linux i32: OFF Linux i32em: -xW Mac OS: -xP
-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

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* Options Suggested Replacement

-F	-preprocess_only or -P
-Kpic, -KPIC	-fpic
-prof-format-32	None
-syntax	-syntax-only
-tpp5	None
-tpp6	None
-tpp7	None

Windows* Options Suggested Replacement

/Fm	/map
/G5	None
/G6 (or /GB)	None
/G7	None
/Ge	/Gs0
/Qprof-format-32	None
/ML and /MLd	None
/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* Options

-axi
-axM

-ipo-obj (and -ipo_obj)

-xi

-xM

Windows* Options

/4ccD (and /4ccd)

/Qaxi

/QaxM

/Qipo-obj (and /Qipo_obj)

/Qxi

/QxM

Removed options are not limited to these lists.

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® Itanium® processors and IA-32 processors with Intel® Extended Memory 64 Technology (Intel® EM64T).

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

/

/? compiler option.....	190
/1 compiler option.....	307
/412 compiler option.....	221
/414 compiler option.....	221
/418 compiler option.....	221
/4L132 compiler option.....	105
/4L72 compiler option.....	105
/4L80 compiler option.....	105
/4Na compiler option	49
/4Naltparam compiler option	30
/4Nb compiler option	63
/4Nd compiler option	539
/4Nf compiler option	128
/4Ns compiler option	539
/4R16 compiler option	473
/4R8 compiler option	473
/4Ya compiler option	49
/4Yaltparam compiler option	30
/4Yb compiler option	63
/4Yd compiler option	539
/4Yf compiler option	166
/4Yportlib compiler option.....	22
/4Ys compiler option	538
/align compiler option.....	25
/allow	
fpp_comments compiler option	28
/altparam compiler option	30
/arch compiler option	34
/architecture compiler option	34
/asmattr	
all compiler option	37
machine compiler option	37
none compiler option.....	37
source compiler option.....	37
/asmfile compiler option.....	39
/assume	
bscc compiler option	40
buffered_io compiler option.....	40
byterec1 compiler option	40
cc_omp compiler option	40
dummy_aliases compiler option.....	40
minus0 compiler option	40
none compiler option.....	40
protect_constants compiler option ..	40
source_include compiler option	40
underscore compiler option.....	40

Intel(R) Fortran Compiler Options

/auto compiler option.....	49	little_endian compiler option	70
/automatic compiler option	49	native compiler option	70
/bintext compiler option	56	vaxd compiler option	70
/c compiler option	58	vaxg compiler option	70
/CB compiler option.....	63	/D compiler option.....	76
/ccdefault		/d_lines compiler option	78
default compiler option	61	/dbglibs compiler option	79
fortran compiler option.....	61	/debug	
list compiler option.....	61	extended compiler option	84
/check		full compiler option	84
all compiler option.....	63	minimal compiler option	84
arg_temp_created compiler option .	63	none compiler option.....	84
bounds compiler option	63	partial compiler option.....	84
none compiler option	63	semantic_stepping compiler option.	84
output_conversion compiler option.	63	/debug compiler option	84
unit compiler option.....	63	/debug-parameters	
/check compiler option	63	all compiler option	87
/compile-only compiler option.....	58	none compiler option.....	87
/convert		used compiler option	87
big_endian compiler option.....	70	/define compiler option	76, 88
cray compiler option	70	/dll compiler option.....	90
fdx compiler option	70	/double_size compiler option	91
fgx compiler option	70	/E compiler option.....	99
ibm compiler option	70	/EP compiler option	101

/error_limit compiler option.....	102	libs compiler option	155
/exe compiler option	103	logicals compiler option.....	155
/extend_source compiler option	105	none compiler option.....	155
/extfor compiler option.....	107	/fpscomp compiler option.....	155
/extfpp compiler option.....	108	/FR compiler option	166
/extlnk compiler option	108	/free compiler option	166
/F compiler option.....	111	/G1 compiler option	508
/f66 compiler option.....	112	/G2 compiler option	508
/f77rtl compiler option.....	113	/G2-p9000 compiler option	508
/fast compiler option	118	/G5 compiler option	510
/Fe compiler option.....	103	/G6 compiler option	510
/FI compiler option.....	128	/G7 compiler option	510
/fixed compiler option	128	/GB compiler option	510
/fltconsistency compiler option	129	/Ge compiler option	182
/fp compiler option.....	139	/gen-interfaces compiler option	183
/fpconstant compiler option	149	/Gm compiler option	195
/fpe compiler option.....	151	/Gs compiler option.....	188
/fpp compiler option.....	154	/Gz compiler option.....	195
/fpscomp		/help compiler option	190
all compiler option.....	155	/I compiler option	191, 200
filesfromcmd compiler option.....	155	/iface	
general compiler option	155	cref compiler option.....	195
ioformat compiler option	155	cvf compiler option	195
ldio_spacing compiler option	155	default compiler option.....	195

Intel(R) Fortran Compiler Options

mixed_str_len_arg compiler option	195	/link compiler option	255
stdcall compiler option	195	/logo compiler option	256
stdref compiler option	195	/map compiler option	259
/include compiler option	200	/MD compiler option	264
/inline		/MDd compiler option	264
all compiler option	202	/MDs compiler option	252, 265
manual compiler option	202	/MDsd compiler option	252, 265
none compiler option	202	/MG compiler option	548
size compiler option	202	/ML compiler option	252, 269
speed compiler option	202	/MLd compiler option	252, 269
/intconstant compiler option	219	/module compiler option	271
/integer_size compiler option	221	/MT compiler option	276
/LD compiler option	90	/MTd compiler option	276
/libdir		/MW compiler option	252
all compiler option	250	/MWs compiler option	252
automatic compiler option	250	/names	
none compiler option	250	as_is compiler option	282
user compiler option	250	lowercase compiler option	282
/libdir compiler option	250	uppercase compiler option	282
/libs		/nbs compiler option	40
dll compiler option	252	/nodefine compiler option	76
qwin compiler option	252	/O compiler option	296
qwins compiler option	252	/O1 compiler option	296
static compiler option	252	/O2 compiler option	296

- /O3 compiler option 296
- /Ob compiler option 301
- /object compiler option 303
- /Od compiler option 305
- /Og compiler option 306
- /Op compiler option 129
- /optimize
 - 0 compiler option 296
 - 1 compiler option 296
 - 2 compiler option 296
 - 3 compiler option 296
 - 4 compiler option 296
 - 5 compiler option 296
- /Os compiler option 330
- /Ot compiler option 331
- /Ox compiler option 296
- /Oy compiler option 146
- /P compiler option 357
- /pdbfile compiler option 348
- /preprocess_only compiler option 357
- /Qansi-alias compiler option..... 32
- /Qauto compiler option 49
- /Qauto_scalar compiler option 46
- /Qautodouble compiler option 473
- /Qax compiler option..... 51
- /Qchkstk compiler option 377
- /Qcommon-args compiler option 40
- /Qcomplex-limited-range compiler option
..... 69
- /Qcpp compiler option..... 154
- /Qd_lines compiler option 78
- /Qdps compiler option..... 30
- /Qdyncom compiler option 97
- /Qextend_source compiler option..... 105
- /Qfnsplit compiler option 136
- /Qfpp compiler option 154
- /Qfp-port compiler option 144
- /Qfpstkchk compiler option 163
- /Qftz compiler option..... 170
- /Qglobal-hoist compiler option 185
- /QIA64-fr32 compiler option..... 391
- /Qlfist compiler option 471
- /Qinline-debug-info compiler option.. 204
- /Qinline-factor compiler option 205
- /Qinline-forceinline compiler option .. 207
- /Qinline-max-per-compile compiler
option 209
- /Qinline-max-per-routine compiler option
..... 211
- /Qinline-max-size compiler option 213

/Qinline-max-total-size compiler option	215	/Qopenmp-stubs compiler option.....	315
/Qinline-min-size compiler option	217	/Qoption compiler option.....	432
/Qip compiler option	223	/Qopt-mem-bandwidth compiler option	317
/QIPF-fltacc compiler option.....	229	/Qopt-report compiler option.....	319
/QIPF-flt-eval-method0 compiler option	228	/Qopt-report-file compiler option	320
/QIPF-fma compiler option	231	/Qopt-report-help compiler option.....	321
/QIPF-fp-relaxed compiler option	233	/Qopt-report-level compiler option	323
/QIPF-fp-speculation compiler option.....	235	/Qopt-report-phase compiler option..	325
/Qip-no-inlining compiler option.....	224	/Qopt-report-routine compiler option.	327
/Qip-no-pinlining compiler option.....	226	/Qpad compiler option	336
/Qipo compiler option	237	/Qpad-source compiler option	338
/Qipo-c compiler option	239	/Qparallel compiler option.....	344
/Qipo-S compiler option.....	241	/Qpar-report compiler option.....	340
/Qipo-separate compiler option	243	/Qpar-threshold compiler option	342
/Qivdep-parallel compiler option.....	245	/Qpc compiler option.....	346
/Qlocation compiler option.....	415	/Qprec compiler option	273
/Qlowercase compiler option.....	282	/Qprec-div compiler option.....	351
/Qmap-opts compiler option.....	260	/Qprec-sqrt compiler option	353
/Qnobss-init compiler option.....	286	/Qprefetch compiler option	355
/Qonetrip compiler option.....	307	/Qprof-dir compiler option	360
/Qopenmp compiler option.....	310	/Qprof-file compiler option	362
/Qopenmp-profile compiler option	312	/Qprof-format-32 compiler option.....	364
/Qopenmp-report compiler option	313	/Qprof-gen compiler option	365
		/Qprof-gen-sampling compiler option.....	367

/Qprof-genx compiler option.....	369	/real_size compiler option.....	473
/Qprof-use compiler option.....	371	/recursive compiler option.....	475
/Qrcd compiler option.....	471	/reentrancy	
/Qrct compiler option.....	453	async compiler option	477
/Qsafe-cray-ptr compiler option.....	481	none compiler option.....	477
/Qsave compiler option	483	threaded compiler option	477
/Qscalar-rep compiler option	484	/RTCu compiler option.....	63
/Qsalign compiler option.....	457	/S compiler option.....	480
/Qsox compiler option	491	/source compiler option	489
/Qssp compiler option	493	/stand	
/Qtcheck compiler option.....	504	f90 compiler option.....	495
/Qtrapuv compiler option	169	f95 compiler option.....	495
/Qunroll compiler option	522	none compiler option.....	495
/Quppercase compiler option	282	/stand compiler option	495
/Quse_vcdebug compiler option.....	465	/static compiler option.....	497
/Quse-asm compiler option	526	/syntax-only compiler option	501
/Qvc6 compiler option	466	/Tf compiler option	489
/Qvc7 compiler option	466	/threads compiler option	506
/Qvc7.1 compiler option	466	/traceback compiler option.....	513
/Qvc8 compiler option	466	/u compiler option	518, 520
/Qvec-report compiler option.....	530	/undefine compiler option	520
/Qvms compiler option	532	/us compiler option	40
/Qx compiler option	551	/vms compiler option.....	532
/Qzero compiler option.....	560	/w compiler option.....	538

Intel(R) Fortran Compiler Options

/W0 compiler option	538	/what compiler option.....	547
/W1 compiler option	538	/winapp compiler option.....	548
/warn		/X compiler option.....	200, 553
alignments compiler option.....	538	/Z7 compiler option	84, 178
all compiler option.....	538	/Zd compiler option	84
declarations compiler option.....	538	/Zi compiler option	84, 178
errors compiler option.....	538	/ZI compiler option	250
general compiler option	538	/Zp compiler option	25
ignore_loc compiler option.....	538	/Zs compiler option	501
interfaces compiler option.....	538	1	
none compiler option	538	-1 compiler option	307
stderrs compiler option	538	-132 compiler option	105
truncated_source compiler option	538	6	
uncalled compiler option.....	538	-66 compiler option	112
unused compiler option	538	7	
usage compiler option	538	-72 compiler option	105
/warn compiler option.....	538	8	
/watch		-80 compiler option	105
all compiler option.....	544	A	
cmd compiler option	544	-align compiler option	25
none compiler option	544	-allow fpp_comments compiler option	28
source compiler option	544	-altparam compiler option	30
/watch compiler option	544	-ansi-alias compiler option.....	32
/WB compiler option	546	-arch compiler option	34

-assume 2underscores compiler option 40

-assume bsccl compiler option..... 40

-assume buffered_io compiler option . 40

-assume byterecl compiler option 40

-assume cc_omp compiler option 40

-assume dummy_aliases compiler option..... 40

-assume minus0 compiler option 40

-assume none compiler option 40

-assume protect_constants compiler option..... 40

-assume source_include compiler option 40

-assume underscore compiler option . 40

-auto compiler option..... 49

-autodouble compiler option..... 473

-automatic compiler option 49

-auto-scalar compiler option..... 46

-ax compiler option..... 51

B

-B compiler option 53

-Bdynamic compiler option (Linux* only) 55

-Bstatic compiler option (Linux* only). 57

C

-C compiler option 63

-CB compiler option 63

-ccdefault default compiler option 61

-ccdefault fortran compiler option 61

-ccdefault list compiler option 61

-check all compiler option 63

-check arg_temp_created compiler option 63

-check bounds compiler option 63

-check compiler option..... 63

-check none compiler option..... 63

-check output_conversion compiler option 63

-check unit compiler option 63

CLOMP options

list of..... 619

cluster OpenMP options

list of..... 619

-cm compiler option 539

-common-args compiler option 40

compiler options

affecting DOUBLE PRECISION KIND 91

affecting INTEGER KIND 221

affecting REAL KIND 473

cross-reference tables of 566

deprecated and removed 617

new	4	-debug inline_debug_info compiler option (Linux* only)	82
overview of descriptions of	8	-debug semantic_stepping compiler option (Linux* only)	82
quick reference summary of	566	-debug variable_locations compiler option (Linux* only)	82
-complex-limited-range compiler option	69	-debug-parameters all compiler option	87
-convert big_endian compiler option ..	70	-debug-parameters none compiler option	87
-convert cray compiler option	70	-debug-parameters used compiler option	87
-convert fdx compiler option	70	-double_size compiler option	91
-convert fgx compiler option	70	-dps compiler option	30
-convert ibm compiler option	70	-dryrun compiler option	94
-convert little_endian compiler option.	70	-dynamiclib compiler option (Mac OS* only)	96
-convert native compiler option	70	-dynamic-linker compiler option (Linux* only)	95
-convert vaxd compiler option	70	-dyncom compiler option	97
-convert vaxg compiler option	70	E	
-cpp compiler option.....	154	-E compiler option.....	99
cross reference	566	-e90 compiler option	539
-cxxlib-gcc compiler option.....	74	-e95 compiler option	539
-cxxlib-icc compiler option (Linux* only)	74	-EP compiler option	101
D		-error_limit compiler option	102
-D compiler option	76	-extend_source compiler option	105
-d_lines compiler option	78	F	
-DD compiler option	78	-F compiler option.....	357
-debug extended compiler option (Linux* only).....	82		

- f66 compiler option..... 112
- f77rtl compiler option..... 113
- falias compiler option 117
- fast compiler option..... 118
- fexceptions compiler option 122
- ffnalias compiler option 123
- FI compiler option 128
- finline-functions compiler options.... 125
- finline-limit compiler option..... 127
- fixed compiler option..... 128
- fltconsistency compiler option 129
- fmath-errno compiler option 133
- fminshared compiler option..... 134
- fno-omit-frame-pointer compiler option
..... 146
- fnsplit compiler option (Linux* only) 136
- fp compiler option..... 146
- fpconstant compiler option..... 149
- fpe compiler option..... 151
- fpic compiler option (Linux* only) 153
- fp-model compiler option..... 139
- fpp compiler option..... 154
- fp-port compiler option 144
- fpscomp all compiler option 155
- fpscomp compiler option 155
- fpscomp filesfromcmd compiler option
..... 155
- fpscomp general compiler option 155
- fpscomp ioformat compiler option ... 155
- fpscomp ldio_spacing compiler option
..... 155
- fpscomp libs compiler option 155
- fpscomp logicals compiler option 155
- fpscomp none compiler option 155
- fpstkchk compiler option..... 163
- FR compiler option 166
- fr32 compiler option (Linux* only) 165
- free compiler option..... 166
- fsyntax-only compiler option..... 501
- ftrapuv compiler option 169
- ftz compiler option 170
- funroll-loops compiler option 522
- fverbose-asm compiler option 173
- fvisibility compiler option..... 175
- G**
- g compiler option 178
- gen-interfaces compiler option 183
- global-hoist compiler option..... 185
- H**
- help compiler option 190

I

-l compiler option.....	191, 200
-i2 compiler option.....	221
-i4 compiler option.....	221
-i8 compiler option.....	221
-i-dynamic compiler option	192
-implicitnone compiler option.....	539
-inline-debug-info compiler option (Linux* only).....	204
-inline-factor compiler option	205
-inline-forceinline compiler option.....	207
-inline-max-per-compile compiler option	209
-inline-max-per-routine compiler option	211
-inline-max-size compiler option.....	213
-inline-max-total-size compiler option.....	215
-inline-min-size compiler option.....	217
-intconstant compiler option	219
-integer_size compiler option	221
introduction to Compiler Options.....	1
-ip compiler option.....	223
-IPF-fltacc compiler option (Linux* only)	229
-IPF-flt-eval-method0 compiler option (Linux* only).....	228

-IPF-fma compiler option (Linux* only)	231
-IPF-fp-relaxed compiler option (Linux* only)	233
-IPF-fp-speculation compiler option (Linux* only)	235
-ip-no-inlining compiler option.....	224
-ip-no-pinlining compiler option.....	226
-ipo compiler option	237
-ipo-c compiler option	239
-ipo-S compiler option.....	241
-ipo-separate compiler option	243
-i-static compiler option.....	193
-isystem compiler option	244
-ivdep-parallel compiler option (Linux* only)	245

K

-Kpic compiler option (Linux* only) ...	153
---	-----

L

-l compiler option	247
--------------------------	-----

Linux* compiler options

-1307	
-132.....	105
-66.....	112
-72.....	105
-80.....	105

-align.....	25	-cxxlib-icc	74
-allow fpp_comments.....	28	-D	76
-altparam	30	-d_lines	78
-ansi-alias	32	-DD.....	78
-arch	34	-debug.....	82
-assume.....	40	-debug-parameters	87
-auto	49	-double_size.....	91
-auto_scalar.....	46	-dps	30
-autodouble.....	473	-dryrun.....	94
-automatic.....	49	-dynamic-linker.....	95
-ax.....	51	-dyncom	97
-B	53	-E	99
-Bdynamic.....	55	-e90	538
-Bstatic.....	57	-e95.....	538
-C.....	63	-EP	101
-CB	63	-error_limit.....	102
-ccdefault.....	61	-extend_source	105
-check.....	63	-F357	
-cm.....	538	-f66.....	112
-common-args	40	-f77rtl.....	113
-complex-limited-range	69	-falias	117
-convert.....	70	-fast	118
-cpp.....	154	-fexceptions.....	122
-cxxlib-gcc.....	74	-ffnalias	123

Intel(R) Fortran Compiler Options

-FI	128	-funroll-loops	522
-finline-functions	125	-fverbose-asm	173
-finline-limit	127	-fvisibility	175
-fixed.....	128	-g 178	
-fltconsistency.....	129	-gen-interfaces	183
-fmath-errno.....	133	-global-hoist	185
-fminshared.....	134	-help	190
-fno-omit-frame-pointer.....	146	-l 191, 200	
-fnsplit.....	136	-i2	221
-fp	146	-i4	221
-fpconstant.....	149	-i8	221
-fpe	151	-i-dynamic	192
-fpic.....	153	-inline-debug-info	204
-fp-model	139	-inline-factor	205
-fpp	154	-inline-forceinline	207
-fp-port.....	144	-inline-max-per-compile	209
-fpscomp.....	155	-inline-max-per-routine	211
-fpstkchk	163	-inline-max-size	213
-FR.....	166	-inline-max-total-size	215
-fr32	165	-inline-min-size	217
-free	166	-intconstant	219
-fsyntax-only	501	-integer_size	221
-ftrapuv	169	-ip	223
-ftz.....	170	-IPF-fltacc.....	229

-IPF-flt-eval-method0.....	228	-mrelax	275
-IPF-fma.....	231	-mtune	278
-IPF-fp-relaxed.....	233	-names	282
-IPF-fp-speculation	235	-nbs	40
-ip-no-inlining	224	-nobss-init	286
-ip-no-pinlining	226	-nodefaultlibs.....	287
-ipo.....	237	-nofor_main	289
-ipo-c.....	239	-nostartfiles	290
-ipo-S	241	-nostdinc	553
-ipo-separate	243	-nostdlib	292
-i-static	193	-nus	40
-isystem	244	-o294	
-ivdep-parallel	245	-O	296
-Kpic	153	-O0	296
-L248		-O1	296
-logo.....	256	-O2	296
-lowercase	282	-O3	296
-map-opts	260	-Ob	301
-mcmodel.....	262	-onetrip	307
-mieee-fp	129	-openmp	310
-mixed_str_len_arg.....	195	-openmp-profile	312
-module.....	271	-openmp-report	313
-mp	129	-openmp-stubs	315
-mp1	273	-opt-mem-bandwidth	317

Intel(R) Fortran Compiler Options

-opt-report.....	319	-prof-genx.....	369
-opt-report-file	320	-prof-use.....	371
-opt-report-help.....	321	-Qinstall.....	401
-opt-report-level	323	-Qlocation.....	415
-opt-report-phase.....	325	-Qoption	432
-opt-report-routine.....	327	-qp	334
-p 334, 357		-r16.....	473
-pad	336	-r8.....	473
-pad-source	338	-rcd.....	471
-parallel.....	344	-real_size	473
-par-report.....	340	-recursive	475
-par-threshold	342	-reentrancy	477
-pc.....	346	-RTCu	63
-pg	334	-S	480
-prec-div.....	351	-safe-cray-ptr.....	481
-prec-sqrt	353	-save	483
-prefetch	355	-scalar-rep.....	484
-preprocess_only	357	-shared.....	486
-print-multi-lib.....	359	-shared-libcxa	487
-prof-dir	360	-sox	491
-prof-file	362	-ssp	493
-prof-format-32.....	364	-stand	495
-prof-gen	365	-static	497
-prof-gen-sampling	367	-static-libcxa	498

-std.....	495	-w	538
-std90.....	495	-W0	538
-std95.....	495	-W1	538
-syntax	501	-Wa	537
-syntax-only	501	-warn	538
-T503		-watch	544
-tcheck	504	-WB	546
-Tf	489	-what	547
-threads	506	-WI.....	549
-tpp1	508	-Wp	550
-tpp2	508	-X	200, 553
-tpp5	510	-Xlinker	555
-tpp6	510	-y 501	
-tpp7	510	-zero.....	560
-traceback.....	513	-Zp.....	25
-tune	515	-logo compiler option	256
-U	520, 538	-lowercase compiler option	282
-unroll.....	522	M	
-uppercase.....	282	Mac OS* compiler options	
-us.....	40	-1 307	
-use-asm.....	526	-132.....	105
-V	527	-66.....	112
-vec-report	530	-72.....	105
-vms.....	532	-80.....	105

Intel(R) Fortran Compiler Options

-align.....	25	-DD.....	78
-allow fpp_comments.....	28	-debug-parameters	87
-altparam	30	-double_size.....	91
-ansi-alias	32	-dps	30
-arch	34	-dryrun.....	94
-assume.....	40	-dynamiclib.....	96
-auto	49	-dyncom	97
-auto_scalar.....	46	-E	99
-autodouble.....	473	-e90.....	538
-automatic.....	49	-e95.....	538
-B.....	53	-EP	101
-C.....	63	-error_limit.....	102
-CB	63	-extend_source	105
-ccdefault.....	61	-F357	
-check.....	63	-f66.....	112
-cm.....	538	-f77rtl.....	113
-common-args	40	-falias	117
-complex-limited-range.....	69	-fast.....	118
-convert.....	70	-fexceptions.....	122
-cpp.....	154	-ffnalias	123
-cxxlib-gcc.....	74	-FI.....	128
-cxxlib-icc.....	74	-finline-functions.....	125
-D.....	76	-finline-limit.....	127
-d_lines.....	78	-fixed	128

-fltconsistency.....	129	-l 191, 200	
-fmath-errno.....	133	-i2	221
-fminshared.....	134	-i4	221
-fno-omit-frame-pointer.....	146	-i8	221
-fp	146	-i-dynamic	192
-fpconstant.....	149	-inline-factor	205
-fpe	151	-inline-forceinline	207
-fp-model	139	-inline-max-per-compile	209
-fpp	154	-inline-max-per-routine	211
-fp-port.....	144	-inline-max-size	213
-fpscomp.....	155	-inline-max-total-size	215
-fstkchk	163	-inline-min-size	217
-FR.....	166	-intconstant	219
-free	166	-integer_size	221
-fsyntax-only	501	-ip	223
-ftrapuv	169	-ip-no-inlining	224
-ftz.....	170	-ip-no-pinlining	226
-funroll-loops.....	522	-ipo	237
-fverbose-asm.....	173	-ipo-c	239
-fvisibility	175	-ipo-S	241
-g 178		-ipo-separate	243
-gen-interfaces.....	183	-i-static	193
-global-hoist.....	185	-isystem.....	244
-help.....	190	-l 247	

Intel(R) Fortran Compiler Options

-logo.....	256	-O3	296
-lowercase	282	-Ob	301
-map-opts	260	-onetrip	307
-mcmodel.....	262	-openmp	310
-mieee-fp	129	-openmp-report	313
-mixed_str_len_arg.....	195	-openmp-stubs	315
-module.....	271	-opt-report	319
-mp	129	-opt-report-file	320
-mp1	273	-opt-report-help	321
-mtune	278	-opt-report-level.....	323
-names.....	282	-opt-report-phase	325
-nbs.....	40	-opt-report-routine	327
-nobss-init.....	286	-p 334, 357	
-nodefaultlibs	287	-pad.....	336
-nofor_main	289	-pad-source	338
-nostartfiles	290	-parallel	344
-nostdinc.....	553	-par-report	340
-nostdlib	292	-par-threshold.....	342
-nus.....	40	-pc	346
-o 294		-pg.....	334
-O.....	296	-prec-div	351
-O0.....	296	-prec-sqrt.....	353
-O1.....	296	-preprocess_only	357
-O2.....	296	-print-multi-lib	359

-prof-dir	360	-std90	495
-prof-file	362	-std95	495
-prof-gen	365	-syntax-only	501
-prof-gen-sampling	367	-Tf	489
-prof-genx	369	-threads	506
-prof-use	371	-traceback	513
-Qinstall	401	-tune	515
-Qlocation	415	-U	520, 538
-Qoption	432	-unroll	522
-qp	334	-uppercase	282
-r16	473	-us	40
-r8	473	-use-asm	526
-rcd	471	-v	527
-real_size	473	-vec-report	530
-recursive	475	-vms	532
-reentrancy	477	-w	538
-RTCu	63	-W0	538
-S	480	-W1	538
-safe-cray-ptr	481	-Wa	537
-save	483	-warn	538
-scalar-rep	484	-watch	544
-sox	491	-WB	546
-stand	495	-what	547
-std	495	-WI	549

-Wp	550
-X	200, 553
-Xlinker	555
-y 501	
-zero	560
-Zp	25
-map-opts compiler option.....	260
-mcmodel=large compiler option	262
-mcmodel=medium compiler option .	262
-mcmodel=small compiler option	262
-mieee-fp compiler option.....	129
-mixed_str_len_arg compiler option .	195
-module compiler option.....	271
-mp compiler option.....	129
-mp1 compiler option.....	273
-mrelax compiler option (Linux* only)	275
-mtune=itanium compiler option (Linux* only)	278
-mtune=itanium2 compiler option (Linux* only)	278
-mtune=itanium2-p9000 compiler option (Linux* only).....	278
-mtune=pentium compiler option.....	278
-mtune=pentium2 compiler option....	278
-mtune=pentium4 compiler option....	278

-mtune=pentium-mmx compiler option	278
-mtune=pentiumpro compiler option .	278

N

-names as_is compiler option	282
-names lowercase compiler option ...	282
-names uppercase compiler option ..	282
-nbs compiler option	40
-nobss-init compiler option.....	286
-no-cpprt compiler option	74
-nodefaultlibs compiler option	287
-nodefine compiler option	76, 88
-nofor_main compiler option	289
-nostartfiles compiler option.....	290
-nostdinc compiler option.....	553
-nostdlib compiler option.....	292
-nus compiler option	40

O

-o compiler option	294
-O compiler option	296
-O0 compiler option	296
-O1 compiler option	296
-O2 compiler option	296
-O3 compiler option	296
-Ob compiler option	301

-onetrip compiler option..... 307

-openmp compiler option..... 310

OpenMP* options

 related cluster options 619

-openmp-profile compiler option (Linux* only)..... 312

-openmp-report compiler option 313

-openmp-stubs compiler option..... 315

-opt-mem-bandwidth compiler option (Linux* only)..... 317

-opt-report compiler option..... 319

-opt-report-file compiler option 320

-opt-report-help compiler option 321

-opt-report-level compiler option 323

-opt-report-phase compiler option 325

-opt-report-routine compiler option... 327

P

-p compiler option..... 334, 357

-pad compiler option..... 336

-pad-source compiler option..... 338

-parallel compiler option 344

-par-report compiler option..... 340

-par-threshold compiler option 342

-pc compiler option..... 346

-pg compiler option..... 334

-prec-div compiler option 351

-prec-sqrt compiler option 353

-prefetch compiler option 355

-preprocess_only compiler option..... 357

-print-multi-lib compiler option 359

-prof-dir compiler option..... 360

-prof-file compiler option 362

-prof-format-32 compiler option (Linux* only) 364

-prof-gen compiler option..... 365

-prof-gen-sampling compiler option .. 367

-prof-genx compiler option 369

-prof-use compiler option..... 371

Q

-Qinstall compiler option 401

-Qlocation compiler option 415

-Qoption compiler option 432

-qp compiler option 334

R

-r16 compiler option..... 473

-r8 compiler option 473

-rcd compiler option 471

-real_size compiler option..... 473

-recursive compiler option 475

-reentrancy async compiler option.... 477

-reentrancy none compiler option..... 477

-reentrancy threaded compiler option
..... 477

-RTCu compiler option 63

S

-S compiler option 480

-safe-cray-ptr compiler option 481

-save compiler option..... 483

-scalar-rep compiler option..... 484

-shared compiler option (Linux* only)486

-shared-libcxa compiler option (Linux*
only)..... 487

-sox compiler option..... 491

-ssp compiler option (Linux* only) 493

-stand compiler option..... 495

-stand f90 compiler option..... 495

-stand f95 compiler option..... 495

-stand none compiler option..... 495

-static compiler option (Linux* only) . 497

-static-libcxa compiler option (Linux*
only)..... 498

-std compiler option..... 495

-std90 compiler option..... 495

-std95 compiler option..... 495

-syntax compiler option (Linux* only) 501

-syntax-only compiler option 501

T

-T compiler option (Linux* only) 503

-tcheck compiler option (Linux* only) 504

-Tf compiler option 489

-threads compiler option 506

-tpp1 compiler option (Linux only) 508

-tpp2 compiler option (Linux only) 508

-tpp5 compiler option (Linux* only) ... 510

-tpp6 compiler option (Linux* only) ... 510

-tpp7 compiler option (Linux* only) ... 510

-traceback compiler option 513

-tune pn1 compiler option 515

-tune pn2 compiler option 515

-tune pn3 compiler option 515

-tune pn4 compiler option 515

U

-U compiler option 520, 539

-unroll compiler option 522

-uppercase compiler option 282

-us compiler option 40

-use-asm compiler option 526

V

-vec-report compiler option 530

-vms compiler option 532

W

- w compiler option 539
- W0 compiler option 539
- W1 compiler option 539
- Wa compiler option 537
- warn alignments compiler option 539
- warn all compiler option 539
- warn compiler option..... 539
- warn declarations compiler option .. 539
- warn errors compiler option 539
- warn general compiler option..... 539
- warn ignore_loc compiler option 539
- warn interfaces compiler option 539
- warn none compiler option..... 539
- warn stderrs compiler option..... 539
- warn truncated_source compiler option
..... 539
- warn uncalled compiler option 539
- warn unused compiler option 539
- warn usage compiler option 539
- watch all compiler option..... 544
- watch cmd compiler option 544
- watch compiler option 544
- watch none compiler option 544
- watch source compiler option 544
- WB compiler option 546
- what compiler option 547
- Windows* compiler options
 - /? 190
 - /1 307
 - /4I2 221
 - /4I4 221
 - /4I8 221
 - /4L132 105
 - /4L72 105
 - /4L80 105
 - /4Na 49
 - /4Naltparam 30
 - /4Nb 63
 - /4Nd 538
 - /4Nf 128
 - /4Ns..... 538
 - /4R16 473
 - /4R8 473
 - /4Ya..... 49
 - /4Yaltparam..... 30
 - /4Yb..... 63
 - /4Yd..... 538
 - /4Yf..... 166

Intel(R) Fortran Compiler Options

/4Yportlib	22	/debug	84
/4Ys	538	/debug-parameters	87
/align	25	/define	76, 88
/allow		/dll	90
fpp_comments	28	/double_size	91
/altparam	30	/E99	
/arch	34	/EP	101
/architecture	34	/error_limit	102
/asmattr	37	/exe	103
/asmfile	39	/extend_source	105
/assume	40	/extfor	107
/auto	49	/extfpp	108
/automatic	49	/extlnk	109
/bintext	56	/F 111	
/C	63	/f66	112
/CB	63	/f77rtl	113
/ccdefault	61	/fast	118
/check	63	/Fe	103
/cm	128	/FI	128
/compile-only	58	/fixed	128
/convert	70	/fltconsistency	129
/D	76, 88	/fp	139
/d_lines	78	/fpconstant	149
/dbglibs	79	/fpe	151

/fpp.....	154	/libs.....	252
/fpscomp.....	155	/link.....	255
/FR.....	166	/logo.....	256
/free.....	166	/map.....	259
/G1.....	508	/MD.....	264
/G2.....	508	/MDd.....	264
/G2-p9000.....	508	/MDsd.....	252, 265
/G5.....	510	/MG.....	548
/G6.....	510	/ML.....	252, 269
/G7.....	510	/MLd.....	252, 269
/GB.....	510	/module.....	271
/Ge.....	182	/MT.....	276
/gen-interfaces.....	183	/MTd.....	276
/Gm.....	195	/MW.....	252
/Gs.....	188	/MWs.....	252
/Gz.....	195	/names.....	282
/help.....	190	/nbs.....	40
/I 191, 200		/O.....	296
/iface.....	195	/O1.....	296
/include.....	200	/O2.....	296
/inline.....	202	/O3.....	296
/intconstant.....	219	/Ob.....	301
/LD.....	90	/object.....	303
/libdir.....	250	/Od.....	305

Intel(R) Fortran Compiler Options

/Og.....	306	/Qfpp	154
/Op.....	129	/Qfp-port.....	144
/optimize	296	/Qfpstkchk	163
/Os.....	330	/Qftz	170
/Ot.....	331	/Qglobal-hoist.....	185
/Ox.....	296	/QIA64-fr32	391
/Oy	146	/Qlfist.....	471
/P357		/Qinline-debug-info.....	204
/pdbfile	348	/Qinline-factor.....	205
/preprocess_only	357	/Qinline-forceinline	207
/Qansi-alias.....	32	/Qinline-max-per-compile	209
/Qauto.....	49	/Qinline-max-per-routine	211
/Qauto_scalar	46	/Qinline-max-size	213
/Qautodouble	473	/Qinline-max-total-size	215
/Qax	51	/Qinline-min-size	217
/Qchkstk.....	377	/Qip	223
/Qcommon-args.....	40	/QIPF-fltacc	229
/Qcomplex-limited-range	69	/QIPF-flt-eval-method0.....	228
/Qcpp	154	/QIPF-fma	231
/Qd_lines	78	/QIPF-fp-relaxed	233
/Qdps	30	/QIPF-fp-speculation	235
/Qdyncom	97	/Qip-no-inlining.....	224
/Qextend_source	105	/Qip-no-pinlining.....	226
/Qfnsplit	136	/Qipo	237

/Qipo-c	239	/Qpar-report	340
/Qipo-S	241	/Qpar-threshold	342
/Qipo-separate	243	/Qpc	346
/Qivdep-parallel	245	/Qprec	273
/Qlocation	415	/Qprec-div	351
/Qlowercase	282	/Qprec-sqrt	353
/Qmap-opts	260	/Qprefetch	355
/Qnobss-init	286	/Qprof-dir	360
/Qonetrip	307	/Qprof-file	362
/Qopenmp	310	/Qprof-format-32	364
/Qopenmp-profile	312	/Qprof-gen	365
/Qopenmp-report	313	/Qprof-gen-sampling	367
/Qopenmp-stubs	315	/Qprof-genx	369
/Qoption	432	/Qprof-use	371
/Qopt-mem-bandwidth	317	/Qrcd	471
/Qopt-report	319	/Qrct	453
/Qopt-report-file	320	/Qsafe-cray-ptr	481
/Qopt-report-help	321	/Qsave	483
/Qopt-report-level	323	/Qscalar-rep	484
/Qopt-report-phase	325	/Qsalign	457
/Qopt-report-routine	327	/Qsox	491
/Qpad	336	/Qssp	493
/Qpad-source	338	/Qtcheck	504
/Qparallel	344	/Qtrapuv	169

Intel(R) Fortran Compiler Options

/Qunroll	522	/u 518, 520	
/Quppercase	282	/undefine	520
/Quse_vcdebug	465	/us	40
/Quse-asm	526	/vms	532
/Qvc6	466	/w	538
/Qvc7	466	/W0.....	538
/Qvc7.1	466	/W1.....	538
/Qvc8	466	/warn	538
/Qvec-report.....	530	/watch.....	544
/Qvms	532	/WB	546
/Qx	551	/what	547
/Qzero.....	560	/winapp.....	548
/real_size	473	/X200, 553	
/recursive	475	/Z7	84, 178
/reentrancy.....	477	/Zd.....	84
/RTCu	63	/Zi	84, 178
/S480		/Zl	250
/source	489	/Zp.....	25
/stand.....	495	/Zs	501
/static	497	-Wl compiler option.....	549
/syntax-only	501	-Wp compiler option.....	550
/Tf	489	x	
/threads.....	506	-X compiler option.....	200, 553
/traceback	513	-x compiler option	551

-Xlinker compiler option..... 555

Y

-y compiler option..... 501

Z

-zero compiler option560

-Zp compiler option.....25