

NAME

eko - The complete list of options and flags for the QLogic PathScale(TM) Compiler Suite

CG, INLINE, IPA, LANG, LNO, OPT, TENV, WOPT – other major topics covered

DESCRIPTION

This man page describes the various flags available for use with the QLogic PathScale **pathcc**, **pathCC**, and **pathf95** compilers.

OPTIMIZATION FLAGS

Some suboptions either enable or disable the feature. To enable a feature, either specify only the suboption name or specify **=1**, **=ON**, or **=TRUE**. Disabling a feature, is accomplished by adding **=0**, **=OFF**, or **=FALSE**. These values are insensitive to case: 'on' and 'ON' mean the same thing. Below, **ON** and **OFF** are used to indicate the enabling or disabling of a feature.

Many options have an opposite ("no-") counterpart. This is represented as **[no-]** in the option description and if used, will turn off or prevent the action of the option. If no **[no-]** is shown, there is no opposite option to the listed option.

-### Like the **-v** option, only nothing is run and args are quoted.

-A pred=ans

Make an assertion with the predicate 'pred' and answer 'ans'. The **-pred=ans** form cancels an assertion with predicate 'pred' and answer 'ans'.

-alignN Align data on common blocks to specified boundaries. The alignN specifications are as follows:

Option Action

-align8 Align data in common blocks to 8-bit boundaries.

-align16 Align data in common blocks to 16-bit boundaries.

-align32 Align data in common blocks 32-bit boundaries.

-align64 Align data in common blocks to 64-bit boundaries. This is the default.

-align128

Align data in common blocks to 128-bit boundaries.

When an alignment is specified, objects smaller than the specification are aligned on boundaries that correspond to their sizes. For example, when **align64** is specified, 32-bit and larger objects are aligned on 32-bit boundaries; 16-bit and larger objects are aligned on 16-bit boundaries; and 8-bit and larger objects are aligned on 8-bit boundaries.

-ansi (For **Fortran**) Generate messages about constructs which violate standard Fortran syntax rules and constraints, plus messages about obsolescent and deleted features. This also disables all nonstandard intrinsic functions and subroutines. Specifying **-ansi** in conjunction with **-fullwarn** causes all messages, regardless of level, to be generated.

-ansi (For **C/C++**) Enable pure ANSI/ISO C mode.

-apo This auto-parallelizing option signals the compiler to automatically convert sequential code into parallel code when it is safe and beneficial to do so. The resulting executable can then run faster on a machine with more than one CPU.

-ar Create an archive using **ar(1)** instead of a shared object or executable. The name of the archive is specified by using the **-o** option. Template entities required by the objects being archived are instantiated before creating the archive. The **pathCC** command implicitly passes the **-r** and **-c** options of **ar** to **ar** in addition to the name of the archive and the objects being created. Any other option that can be used in conjunction with the **-c** option of **ar** can be passed to **ar** using **-WR,option_name**.

NOTE: The objects specified with this option must include all of the objects that will be included in the archive. Failure to do so may cause prelinker internal errors. In the following example, **liba.a** is an archive containing only **a.o**, **b.o**, and **c.o**. The **a.o**, **b.o**, and **c.o** objects are prelinked to instantiate any

required template entities, and the **ar -r -c -v lib.a a.o b.o c.o** command is executed. All three objects must be specified with **-ar** even if only **b.o** needs to be replaced in **lib.a**.

```
pathCC -ar -WR,-v -o lib.a a.o b.o c.o
```

See the **ld(1)** man page for more information about shared libraries and archives.

-auto-use *module_name*[,*module_name*] ...

(For **Fortran**) Direct the compiler to behave as if a **USE** *module_name* statement were entered in your Fortran source code for each *module_name*. The **USE** statements are entered in every program unit and interface body in the source file being compiled (for example, **pathf95 -auto-use mpi_interface** or **pathf95 -auto-use shmem_interface**). Using this option can add compiler time in some situations.

-backslash

Treat a backslash as a normal character rather than as an escape character. When this option is used, the preprocessor will not be called.

-C (For **Fortran**) Perform runtime subscript range checking. Subscripts that are out of range cause fatal runtime errors. If you set the **F90_BOUNDS_CHECK_ABORT** environment variable to **YES**, the program aborts.

-C (For **C**) Keep comments after preprocessing.

-c Create an intermediate object file for each named source file, but does not link the object files. The intermediate object file name corresponds to the name of the source file; a **.o** suffix is substituted for the suffix of the source file.

Because they are mutually exclusive, do not specify this option with the **-r** option.

-CG[:...]

The Code Generation option group controls the optimizations and transformations of the instruction-level code generator.

-CG:cflow=(ON|OFF)

OFF disables control flow optimization in the code generation. Default is **ON**.

-CG:cse_regs=N

When performing common subexpression elimination during code generation, assume there are **N** extra **integer** registers available over the number provided by the CPU. **N** can be positive, zero, or negative. The default is positive infinity. See also **-CG:sse_cse_regs**.

-CG:gcm=(ON|OFF)

Specifying **OFF** disables the instruction-level global code motion optimization phase. The default is **ON**.

-CG:load_exe=N

Specify the threshold for subsuming a memory load operation into the operand of an arithmetic instruction. The value of **0** turns off this subsumption optimization. If **N** is 1, this subsumption is performed only when the result of the load has only one use. This subsumption is not performed if the number of times the result of the load is used exceeds the value **N**, a non-negative integer. The default value varies based on processor target and source language.

-CG:local_fwd_sched=(ON|OFF)

Change the instruction scheduling algorithm to work forward instead of backward for the instructions in each basic block. The default is **OFF** for 64-bit ABI, and **ON** for 32-bit ABI.

-CG:movnti=N

Convert ordinary stores to non-temporal stores when writing memory blocks of size larger than **N** KB. When **N** is set to **0**, this transformation is avoided. The default value is **1000 (KB)**.

-CG:p2align=(ON|OFF)

Align loop heads to 64-byte boundaries. The default is **OFF**.

- CG:p2align_freq=N**
Align branch targets based on execution frequency. This option is meaningful only under feed-back-directed compilation. The default value **N=0** turns off the alignment optimization. Any other value specifies the frequency threshold at or above which this alignment will be performed by the compiler.
- CG:prefer_legacy_regs=(ON|OFF)**
Tell the local register allocator to use the first 8 integer and SSE registers whenever possible (%rax-%rbp, %xmm0-%xmm7). Instructions using these registers have smaller instruction sizes. The default is **OFF**.
- CG:prefetch=(ON|OFF)**
Enable generation of prefetch instructions in the code generator. The default is **ON**. (**-CG:prefetch=OFF** and **-LNO:prefetch=0** both suppress the generation of prefetch instructions, but **-LNO:prefetch=0** also affects LNO optimizations that depend on prefetch.)
- CG:sse_cse_regs=N**
When performing common subexpression elimination during code generation, assume there are **N** extra SSE registers available over the number provided by the CPU. **N** can be positive, zero, or negative. The default is positive infinity. See also **-CG:cse_regs**.
- CG:use_prefetchnta=(ON|OFF)**
Prefetch when data is non-temporal at all levels of the cache hierarchy. This is for data streaming situations in which the data will not need to be re-used soon. The default is **OFF**.
- CG:use_test=(ON|OFF)**
Make the code generator use the TEST instruction instead of CMP. See Opteron's instruction description for the difference between these two instructions. The default is **OFF**.
- clist** (C only) Enable the C listing. Specifying **-clist** is the equivalent of specifying **-CLIST:=ON**.
- CLIST: ...**
(C only) Control emission of the compiler's internal program representation back into C code, after IPA inlining and loop-nest transformations. This is a diagnostic tool, and the generated C code may not always be compilable. The generated C code is written to two files, a header file containing file-scope declarations, and a file containing function definitions. With the exception of **-CLIST:=OFF**, any use of this option implies **-clist**. The individual controls in this group are as follows:
 - =(ON|OFF)**
Enable the C listing. This option is implied by any of the others, but may be used to enable the listing when no other options are required. For example, specifying **-CLIST:=ON** is the equivalent of specifying **-clist**.
 - dotc_file=filename**
Write the program units into the specified file, *filename*. The default source file name has the extension **.w2c.c**.
 - doth_file=filename**
Specify the file into which file-scope declarations are deposited. Defaults to the source file name with the extension **.w2c.h**.
 - emit_pfetch[=(ON|OFF)]**
Display prefetch information as comments in the transformed source. If **ON** or **OFF** is not specified, the default is **OFF**.
 - linelength=N**
Set the maximum line length to *N* characters. The default is unlimited.
 - show[=(ON|OFF)]**
Print the input and output file names to stderr. If **ON** or **OFF** is not specified, the default is **ON**.
- colN** (**Fortran** only) Specify the line width for fixed-format source lines. Specify **72**, **80**, or **120** for *N* (-col72, -col80, or -col120). By default, fixed-format lines are 72 characters wide. Specifying **-col120** implies **-extend-source** and recognizes lines up to 132 characters wide. For more information on specifying line

length, see the **-extend-source** and **-noextend-source** options.

-copyright

Show the copyright for the compiler being used.

-cpp Run the preprocessor, **cpp**, on all input source files, regardless of suffix, before compiling. This preprocessor automatically expands macros outside of preprocessor statements.

The default is to run the C preprocessor (**cpp**) if the input file ends in a **.F** or **.F90** suffix.

For more information on controlling preprocessing, see the **-ftpp**, **-E**, and **-nocpp** options. For information on enabling macro expansion, see the **-macro-expand** option. By default, no preprocessing is performed on files that end in a **.f** or **.f90** suffix.

-d-lines (**Fortran** only) Compile lines with a D in column 1.

-Dvar=[def][,var=[def] . . .]

Define variables used for source preprocessing as if they had been defined by a **#define** directive. If no *def* is specified, **1** is used. For information on undefining variables, see the **-Uvar** option.

-default64

(For **Fortran** only) Set the sizes of default integer, real, logical, and double precision objects. This option is a synonym for the pair of options: **-r8 -i8**. Calling a routine in a specialized library, such as **SCSL**, requires that its 64-bit entry point be specified when 64-bit data are used. Similarly, its 32-bit entry point must be specified when 32-bit data are used.

-dumpversion

Show the version of the compiler being used and nothing else.

-E Run only the source preprocessor files, without considering suffixes, and write the result to **stdout**. This option overrides the **-nocpp** option. The output file contains line directives. To generate an output file without line directives, see the **-P** option. For more information on controlling source preprocessing, see the **-cpp**, **-ftpp**, **-macro-expand**, and **-nocpp** options.

-extend-source

(For **Fortran** only) Specify a 132-character line length for fixed-format source lines. By default, fixed-format lines are 72 characters wide. For more information on controlling line length, see the **-coln** option.

-fb-create <path>

Used to specify that an instrumented executable program is to be generated. Such an executable is suitable for producing feedback data files with the specified prefix for use in feedback-directed compilation (FDO). The commonly used prefix is <fbdata>. This is **OFF** by default.

-fb-opt <prefix for feedback data files>

Used to specify feedback-directed compilation (FDO) by extracting feedback data from files with the specified prefix, which were previously generated using **-fb-create**. The commonly used prefix is "fbdata". The same optimization flags must have been used in the **-fb-create** compile. Feedback data files created from executables compiled with different optimization flags will give checksum errors. FDO is **OFF** by default.

-fb-phase=(0,1,2,3,4)

Used to specify the compilation phase at which instrumentation for the collection of profile data is performed, so is useful only when used with **-fb-create**. The values must be in the range 0 to 4. The default value is 0, and specifies the earliest phase for instrumentation, which is after the front-end processing.

-f[no-]check-new

(For **C++** only) Check the result of new for NULL. When **-fno-check-new** is used, the compiler will not check the result of an operator of NULL.

-fe Stop after the front-end is run.

-f[no-]unwind-tables

-funwind-tables emits unwind information. **-fno-unwind-tables** tells the compiler never to emit any unwind information. This is the default. Flags to enable exception handling automatically enable **-funwind-tables**.

-f[no-]fast-math

-ffast-math improves FP speed by relaxing ANSI & IEEE rules. **-ffast-math** is implied by **-Ofast**. **-fno-fast-math** tells the compiler to conform to ANSI and IEEE math rules at the expense of speed. **-ffast-math** implies **-OPT:IEEE_arithmetic=2** **-fno-math-errno**. **-fno-fast-math** implies **-OPT:IEEE_arithmetic=1** **-fmath-errno**.

-f[no-]fast-stdlib

The **-ffast-stdlib** flag improves application performance by generating code to link against special versions of some standard library routines, and linking against the PathScale compiler runtime library. This option is enabled by default.

If **-fno-fast-stdlib** is used during compilation, the compiler will not emit code to link against fast versions of standard library routines. During compilation, **-ffast-stdlib** implies **-OPT:fast_stdlib=on**.

If **-fno-fast-stdlib** is used during linking, the compiler will not link against the PathScale compiler runtime library.

If you link code with **-fno-fast-stdlib** that was not also compiled with this flag, you may see linker errors. Much of the PathScale compiler Fortran runtime is compiled with **-ffast-stdlib**, so it is not advised to link Fortran applications with **-fno-fast-stdlib**.

-ffloat-store

Do not store floating point variables in registers, and inhibit other options that might change whether a floating point value is taken from a register or memory. This option prevents undesirable excess precision on the X87 floating-point unit where all floating-point computations are performed in one precision regardless of the original type. (see **-mx87-precision**). If the program uses floating point values with less precision, the extra precision in the X87 may violate the precise definition of IEEE floating point. **-ffloat-store causes all pertinent immediate computations to be stored to memory to force truncation to lower precision. However, the extra stores will slow down program execution substantially.** **-ffloat-store** has no effect under **-msse2**, which is the default under both **-m64** and **-m32**.

-ffortran-bounds-check

(For Fortran only) Check bounds.

-f[no-]gnu-keywords

(For C/C++ only) Recognize 'typeof' as a keyword. If **-fno-gnu-keywords** is used, do not recognize 'typeof' as a keyword.

-f[no-]implicit-inline-templates

(For C++ only) **-fimplicit-inline-templates** emits code for inline templates instantiated implicitly. **-fno-implicit-inline-templates** tells the compiler to never emit code for inline templates instantiated implicitly.

-f[no-]implicit-templates

(For C++ only) The **-fimplicit-templates** option emits code for non-inline templates instantiated implicitly. With **-fno-implicit-templates** the compiler will not emit code for non-inline templates instantiated implicitly.

-finhibit-size-directive

Do not generate **.size** directives.

-f[no-]inline-functions

(For C/C++ only) **-finline-functions** automatically integrates simple functions into their callers. **-fno-inline-functions** does not automatically integrate simple functions into their callers.

-fabi-version=N

(For C++ only) Use version **N** of the C++ ABI. Version 1 is the version of the C++ ABI that first appeared in G++ 3.2. Version 0 will always be the version that conforms most closely to the C++ ABI specification. Therefore, the ABI obtained using version 0 will change as ABI bugs are fixed. The default is version **1**.

-fixedform

(For Fortran only) Treat all input source files, regardless of suffix, as if they were written in fixed source form (f77 72-column format), instead of F90 free format. By default, only input files suffixed with **.f** or **.F** are assumed to be written in fixed source form.

-fkeep-inline-functions

(For C/C++ only) Generate code for functions even if they are fully inlined.

-FLIST: ...

Invoke the Fortran listing control group, which controls production of the compiler's internal program representation back into Fortran code, after IPA inlining and loop-nest transformations. This is used primarily as a diagnostic tool, and the generated Fortran code may not always compile. With the exception of **-FLIST:=OFF**, any use of this option implies **-flist**. The arguments to the **-FLIST** option are as follows:

Argument

Action

=setting Enable or disable the listing. **setting** can be either **ON** or **OFF**. The default is **OFF**.

This option is enabled when any other **-FLIST** options are enabled, but it can also be used to enable a listing when no other options are enabled.

ansi_format=setting

Set ANSI format. **setting** can be either **ON** or **OFF**. When set to **ON**, the compiler uses a space (instead of tab) for indentation and a maximum of 72 characters per line. The default is **OFF**.

emit_pfetch=setting

Writes prefetch information, as comments, in the transformed source file. **setting** can be either **ON** or **OFF**. The default is **OFF**.

In the listing, **PREFETCH** identifies a prefetch and includes the variable reference (with an offset in bytes), an indication of read/write, a stride for each dimension, and a number in the range from 1 (low) to 3 (high), which reflects the confidence in the prefetch analysis. Prefetch identifies the reference(s) being prefetched by the **PREFETCH** descriptor. The comments occur after a read/write to a variable and note the identifier of the **PREFETCH**-spec for each level of the cache.

ftn_file=file

Write the program to *file*. By default, the program is written to *file.w2f.f*.

linelength=N

Set the maximum line length to *N* characters.

show=setting

Write the input and output filenames to **stderr**. **setting** can be either **ON** or **OFF**. The default is **ON**.

-flist Invoke all Fortran listing control options. The effect is the same as if all **-FLIST** options are enabled.

-fms-extensions

(For C/C++ only) Accept broken MFC extensions without warning.

-fno-asm

(For C/C++ only) Do not recognize the 'asm' keyword.

-fno-builtin

(For C/C++ only) Do not recognize any built in functions.

-fno-common

(For C/C++ only) Use strict ref/def initialization model.

-f[no-]exceptions

(For C++ only) **-fexceptions** enables exception handling. This is the default. **-fno-exceptions** disables exception handling. This option has a subset of the effects of **-fno-gnu-exceptions**. Hence, it can be used on some C++ applications, on which **-fno-gnu-exceptions** cannot be applied.

-f[no-]fast-math

-ffast-math improves FP speed by relaxing ANSI & IEEE rules. **-fno-fast-math** tells the compiler to conform to ANSI and IEEE math rules at the expense of speed.

-f[no-]gnu-exceptions

(For C++ only) **-fgnu-exceptions** enables exception handling, and is equivalent to **-fexceptions**. This is the default. **-fno-gnu-exceptions** disables exception handling, and is equivalent to GNU option **-fno-exceptions**.

-fno-ident

Ignore #ident directives.

-fno-math-errno

Do not set ERRNO after calling math functions that are executed with a single instruction, e.g. **sqrt**. A program that relies on IEEE exceptions for math error handling may want to use this flag for speed while maintaining IEEE arithmetic compatibility. This is implied by **-Ofast**. The default is **-fmath-errno**.

-f[no-]signed-char

(For C/C++ only) **-fsigned-char** makes 'char' signed by default. **-fno-signed-char** makes 'char' unsigned by default.

-fpack-struct

(For C/C++ only) Pack structure members together without holes.

-f[no-]permissive

-fpermissive will downgrade messages about non-conformant code to warnings. **-fno-permissive** keeps messages about non-conformant code as errors.

-f[no-]preprocessed

-fpreprocessed tells the preprocessor that input has already been preprocessed. Using **-fno-preprocessed** tells preprocessor that input has not already been preprocessed.

-freeform

(For **Fortran** only) Treats all input source files, regardless of suffix, as if they were written in free source form. By default, only input files suffixed with **.f90** or **.F90** are assumed to be written in free source form.

-f[no-]rtti

(For C++ only) Using **-frtti** will generate runtime type information. The **-fno-rtti** option will not generate runtime type information.

-f[no-]second-underscore

(For **Fortran** only) **-fsecond-underscore** appends a second underscore to symbols that already contain an underscore. **-fno-second-underscore** tells the compiler not to append a second underscore to symbols that already contain an underscore.

-f[no-]signed-bitfields

(For C/C++ only) **-fsigned-bitfields** makes bitfields be signed by default. The **-fno-signed-bitfields** will make bitfields be unsigned by default.

-f[no-]strict-aliasing

(For C/C++ only) **-fstrict-aliasing** tells the compiler to assume strictest aliasing rules. **-fno-strict-aliasing** tells the compiler not to assume strict aliasing rules.

- f[no-]PIC**
-fPIC tells the compiler to generate position independent code, if possible. The default is **-fno-PIC**, which tells the compiler not to generate position independent code.
- fprefix-function-name**
 (For C/C++ only) Add a prefix to all function names.
- fshared-data**
 (For C/C++ only) Mark data as shared rather than private.
- fshort-double**
 (For C/C++ only) Use the same size for double as for float.
- fshort-enums**
 (For C/C++ only) Use the smallest fitting integer to hold enums.
- fshort-wchar**
 (For C/C++ only) Use short unsigned int for wchar_t instead of the default underlying type for the target.
- ftest-coverage**
 Create data files for the **pathcov(1)** code-coverage utility. The data file names begin with the name of your source file:
SOURCENAME.bb
 A mapping from basic blocks to line numbers, which **pathcov** uses to associate basic block execution counts with line numbers.
SOURCENAME.bbg
 A list of all arcs in the program flow graph. This allows **pathcov** to reconstruct the program flow graph, so that it can compute all basic block and arc execution counts from the information in the **SOURCENAME.da** file.
 Use **-ftest-coverage** with **-fprofile-arcs**; the latter option adds instrumentation to the program, which then writes execution counts to another data file:
SOURCENAME.da
 Runtime arc execution counts, used in conjunction with the arc information in the file **SOURCE-NAME.bbg**.
 Coverage data will map better to the source files if **-ftest-coverage** is used without optimization. See the gcc man pages for more information.
- ftpp** Run the Fortran source preprocessor on input Fortran source files before compiling. By default, files suffixed with **.F** or **.F90** are run through the C source preprocessor (**cpp**). Files that are suffixed with **.f** or **.f90** are not run through any preprocessor by default.
 The Fortran source preprocessor does not automatically expand macros outside of preprocessor statements, so you need to specify **-macro-expand** if you want macros expanded.
- fullwarn**
 Request that the compiler generate comment-level messages. These messages are suppressed by default. Specifying this option can be useful during software development.
- f[no-]underscoring**
 (For Fortran only) **-funderscoring** appends underscores to symbols. **-fno-underscoring** tells the compiler not to append underscores to symbols.
- f[no-]unsafe-math-optimizations**
-funsafe-math-optimizations improves FP speed by violating ANSI and IEEE rules. **-fno-unsafe-math-optimizations** makes the compilation conform to ANSI and IEEE math rules at the expense of speed. This option is provided for GCC compatibility and is equivalent to **-OPT:IEEE_arithmetic=3**
-fno-math-errno.

-fuse-cxa-atexit

(For C++ only) Register static destructors with `__cxa_atexit` instead of `atexit`.

-fwritable-strings

(For C/C++ only) Attempt to support writable-strings K&R style C.

-g[N] Specify debugging support and to indicate the level of information produced by the compiler. The supported values for **N** are:

- 0** No debugging information for symbolic debugging is produced. This is the default.
- 1** Produces minimal information, enough for making backtraces in parts of the program that you don't plan to debug. This is also the flag to use if the user wants backtraces but does not want the overhead of full debug information. This flag also causes `--export-dynamic` to be passed to the linker.
- 2** Produces debugging information for symbolic debugging. Specifying `-g` without a debug level is equivalent to specifying `-g2`. If there is no explicit optimization flag specified, the `-O0` optimization level is used in order to maintain the accuracy of the debugging information. If optimization options `-O1`, `-O2`, `-O3` or `-ipa` are explicitly specified, the optimizations are performed accordingly but the accuracy of the debugging cannot be guaranteed.
- 3** Produces additional debugging information for debugging macros.

-gcc (For C/C++ only) Define the `__GNUC__` and other predefined preprocessor macros.

-gnu[N]

(For C/C++ only) Enables the compiler to generate code compatible with the GNU **N** series of compilers, where **N** is either 3 or 4. On systems whose system compiler is GCC 3, the default is `-gnu3`; on GCC 4 systems the default is `-gnu4`. Use `-show-defaults` to display the default.

-GRA:home=(ON|OFF)

Turn off the rematerialization optimization for non-local user variables in the Global Register Allocator. Default is **ON**.

-GRA:optimize_boundary=(ON|OFF)

Allow the Global Register Allocator to allocate the same register to different variables in the same basic-block. Default is **OFF**.

-help List all available options. The compiler is not invoked.

-help: Print list of possible options that contain a given string.

-H Print the name of each header file used.

-Idir Specify a directory to be searched. This is used for the following types of files:

- Files named in **INCLUDE** lines in the Fortran source file that do not begin with a slash (/) character
- Files named in **#include** source preprocessing directives that do not begin with a slash (/) character
- Files specified on Fortran **USE** statements

Files are searched in the following order: first, in the directory that contains the input file; second, in the directories specified by *dir*; and third, in the standard directory, `/usr/include`.

-iN (For Fortran only) Specify the length of default integer constants, default integer variables, and logical quantities. Specify one of the following:

Option Action

- i4** Specifies 32-bit (4 byte-) objects. The default.
- i8** Specifies 64-bit (8 byte-) objects.

-ignore-suffix

Determine the language of the source file being compiled by the command used to invoke the compiler. By default, the language is determined by the file suffixes (`.c`, `.cpp`, `.C`, `.cxx`, `.f`, `.f90`, `.s`). When the `-ignore-suffix` option is specified, the `pathcc` command invokes the C compiler, `pathCC` invokes the C++

compiler, and **pathf95** invokes the Fortran 95 compiler.

-inline Request inline processing.

-INLINE: ...

Specify options for subprogram inlining. may not always compile. With the exception of **-INLINE:=OFF**, any use of this option implies **-inline**.

If you have included inlining directives in your source code, the **-INLINE** option must be specified in order for those directives to be honored.

-INLINE:aggressive=(ON|OFF)

Tell the compiler to be more aggressive about inlining. The default is **-INLINE:aggressive=OFF**.

-INLINE:list=(ON|OFF)

Tell the inliner to list inlining actions as they occur to **stderr**. The default is **-INLINE:list=OFF**.

-INLINE:preempt=(ON|OFF)

Perform inlining of functions marked preemptible in the light-weight inliner. Default is **OFF**. This inlining prevents another definition of such a function, in another DSO, from preempting the definition of the function being inlined.

-ipa Invoke inter-procedural analysis (IPA). Specifying this option is identical to specifying **-IPA** or **-IPA:**. Default settings for the individual IPA suboptions are used.

-IPA: ...

The inter-procedural analyzer option group controls application of inter-procedural analysis and optimization, including inlining, constant propagation, common block array padding, dead function elimination, alias analysis, and others. Specify **-IPA** by itself to invoke the inter-procedural analysis phase with default options. If you compile and link in distinct steps, you must specify at least **-IPA** for the compile step, and specify **-IPA** and the individual options in the group for the link step. If you specify **-IPA** for the compile step, and do not specify **-IPA** for the link step, you will receive an error.

-IPA:addressing=(ON|OFF)

Invoke the analysis of address operator usage. The default is **Off**. **-IPA:alias=ON** is a prerequisite for this option.

-IPA:aggr_cprop=(ON|OFF)

Enable or disable aggressive inter-procedural constant propagation. Setting can be **ON** or **OFF**. This attempts to avoid passing constant parameters, replacing the corresponding formal parameters by the constant values. Less aggressive inter-procedural constant propagation is done by default. The default setting is **ON**.

-IPA:alias=(ON|OFF)

Invoke alias/mod/ref analysis. The default is **ON**.

-IPA:callee_limit=N

Functions whose size exceeds this limit will never be automatically inlined by the compiler. The default is **500**.

-IPA:cgi=(ON|OFF)

Invoke constant global variable identification. This option marks non-scalar global variables that are never modified as constant, and propagates their constant values to all files. Default is **ON**.

-IPA:clone_list=(ON|OFF)

Tell the IPA function cloner to list cloning actions as they occur to **stderr**. The default is **-IPA:clone_list=OFF**.

-IPA:common_pad_size=N

This specifies the amount by which to pad common block array dimensions. By default, an amount is automatically chosen that will improve cache behavior for common block array accesses.

- IPA:cprop=(ON|OFF)**
Turn on or off inter-procedural constant propagation. This option identifies the formal parameters that always have a specific constant value. Default is **ON**. See also **-IPA:aggr_cprop**.
- IPA:ctype=(ON|OFF)**
When **ON**, causes the compiler to generate faster versions of the <ctype.h> macros such as isalpha, isascii, etc. This flag is unsafe both in multi-threaded programs and in all locales other than the 7-bit ASCII (or "C") locale. The default is **OFF**. Do not turn this on unless the program will always run under the 7-bit ASCII (or "C") locale and is single-threaded.
- IPA:depth=N**
Identical to **maxdepth=N**.
- IPA:dfe=(ON|OFF)**
Enable or disable dead function elimination. Removes any functions that are inlined everywhere they are called. The default is **ON**.
- IPA:dve=(ON|OFF)**
Enable or disable dead variable elimination. This option removes variables that are never referenced by the program. Default is **ON**.
- IPA:echo=(ON|OFF)**
Option to echo (to stderr) the compile commands and the final link commands that are invoked from IPA. Default is **OFF**. This option can help monitor the progress of a large system build.
- IPA:field_reorder=(ON|OFF)**
Enable the re-ordering of fields in large structs based on their reference patterns in feedback compilation to minimize data cache misses. The default is **OFF**.
- IPA:forcedepth=N**
This option sets inline depths, directing IPA to attempt to inline all functions at a depth of (at most) **N** in the callgraph, instead of using the default inlining heuristics. This option ignores the default heuristic limits on inlining. Functions at depth 0 make no calls to any sub-functions. Functions only making calls to depth 0 functions are at depth 1, and so on.
- IPA:ignore_lang=(ON|OFF)**
Enable/disable inlining across language boundaries of Fortran on one side, and C/C++ on the other. The compiler may not always be aware of the correct effective language semantics if this optimization is done, making it unsafe in some scenarios. The default is **OFF**.
- IPA:inline=(ON|OFF)**
This option performs inter-file subprogram inlining during the main IPA processing. The default is **ON**. Does not affect the light-weight inliner.
- IPA:keeplight=(ON|OFF)**
This option directs IPA not to send **-keep** to the compiler, in order to save space. The default is **OFF**.
- IPA:linear=(ON|OFF)**
Controls conversion of a multi-dimensional array to a single dimensional (linear) array that covers the same block of memory. When inlining Fortran subroutines, IPA tries to map formal array parameters to the shape of the actual parameter. In the case that it cannot map the parameter, it linearizes the array reference. By default, IPA will not inline such callsites because they may cause performance problems. The default is **OFF**.
- IPA:map_limit=N**
Direct when IPA enables **sp_partition**. **N** is the maximum size (in bytes) of input files mapped before IPA invokes **-IPA:sp_partition**.
- IPA:maxdepth=N**
This option directs IPA to not attempt to inline functions at a depth of more than **N** in the callgraph; where functions that make no calls are at depth 0, those that call only depth 0 functions are at depth 1, and so on. This inlining remains subject to overriding limits on code expansion. Also see **-IPA:forcedepth**,

-IPA:space, and **-IPA:plimit**.

-IPA:max_jobs=N

This option limits the maximum parallelism when invoking the compiler after IPA to (at most) **N** compilations running at once. The option can take the following values:

0 = The parallelism chosen is equal to either the number of CPUs, the number of cores, or the number of hyperthreading units in the compiling system, whichever is greatest.

1 = Disable parallelization during compilation (default)

>1 = Specifically set the degree of parallelism

-IPA:min_hotness=N

When feedback information is available, a call site to a procedure must be invoked with a count that exceeds the threshold specified by **N** before the procedure will be inlined at that call site. The default is 10.

-IPA:multi_clone=N

This option specifies the maximum number of clones that can be created from a single procedure. Default value is **0**. Aggressive procedural cloning may provide opportunities for inter-procedural optimization, but may also significantly increase the code size.

-IPA:node_bloat=N

When this option is used in conjunction with **-IPA:multi_clone**, it specifies the maximum percentage growth of the total number of procedures relative to the original program.

-IPA:plimit=N

This option stops inlining into a specific subprogram once it reaches size **N** in the intermediate representation. Default is **2500**.

-IPA:pu_reorder=(0|1|2)

Control re-ordering the layout of program units based on their invocation patterns in feedback compilation to minimize instruction cache misses. This option is ignored unless under feedback compilation.

0 = Disable procedure reordering. This is the default for non-C++ programs.

1 = Reorder based on the frequency in which different procedures are invoked. This is the default for C++ programs.

2 = Reorder based on caller-callee relationship.

-IPA:relopt=(ON|OFF)

This option enables optimizations similar to those achieved with the compiler options **-O** and **-c**, where objects are built with the assumption that the compiled objects will be linked into a call-shared executable later. The default is **OFF**. In effect, optimizations based on position-dependent code (non-PIC) are performed on the compiled objects.

-IPA:small_pu=N

A procedure with size smaller than **N** is not subjected to the plimit restriction. The default is 30.

-IPA:sp_partition=[setting]

This option enables partitioning for disk/addressing-saving purposes. The default is **OFF**. Mainly used for building very large programs. Normally, partitioning would be done by IPA internally.

-IPA:space=N

Inline until a program expansion of **N%** is reached. For example, **-IPA:space=20** limits code expansion due to inlining to approximately 20%. Default is no limit.

-IPA:specfile=filename

Opens a *filename* to read additional options. The specification file contains zero or more lines with inliner options in the form expected on the command line. The specfile option cannot occur in a specification file, so specification files cannot invoke other specification files.

-IPA:use_intrinsic=(ON|OFF)

Enable/disable loading the intrinsic version of standard library functions. The default is **OFF**.

-isystem dir

Search *dir* for header files, after all directories specified by **-I** but before the standard system directories. Mark it as a system directory, so that it gets the same special treatment as is applied to the standard system directories.

-keep Write all intermediate compilation files. *file.s* contains the generated assembly language code. *file.i* contains the preprocessed source code. These files are retained after compilation is finished. If IPA is in effect and you want to retain *file.s*, you must specify **-IPA:keeplight=OFF** in addition to **-keep**.

-keepdollar

(For **Fortran** only) Treat the dollar sign (\$) as a normal last character in symbol names.

-L directory

In XPG4 mode, changes the algorithm of searching for libraries named in **-L** operands to look in the specified directory before looking in the default location. Directories specified in **-L** options are searched in the specified order. Multiple instances of **-L** options can be specified.

-I library

In XPG4 mode, searches the specified *library*. A library is searched when its name is encountered, so the placement of a **-I** operand is significant.

-LANG: ...

Controls the language option group. The following sections describe the suboptions available in this group.

Argument

Action

copyinout=(ON|OFF)

When an array section is passed as the actual argument in a call, the compiler sometimes copies the array section to a temporary array and passes the temporary array, thus promoting locality in the accesses to the array argument. This optimization is relevant only to Fortran, and this flag controls the aggressiveness of this optimization. The default is **ON** for **-O2** or higher and **OFF** otherwise.

formal_deref_unsafe=(ON|OFF)

Tell the compiler whether it is unsafe to speculate a dereference of a formal parameter in Fortran. The default is **OFF**, which is better for performance.

heap_allocation_threshold=size

Determine heap or stack allocation. If the size of an automatic array or compiler temporary exceeds *size* bytes it is allocated on the heap instead of the stack. If *size* is **-1**, objects are always put on the stack. If *size* is **0**, objects are always put on the heap.

The default is **-1** for maximum performance and for compatibility with previous releases.

IEEE_minus_zero=setting

Enable or disable the **SIGN(3I)** intrinsic function's ability to recognize negative floating-point zero (**-0.0**). Specify either **ON** or **OFF** for *setting*. The default is **OFF**, which suppresses the minus sign. The minus sign is suppressed by default to prevent problems from hardware instructions and optimizations that can return a **-0.0** result from a **0.0** value. To obtain a minus sign (**-**) when printing a negative floating-point zero (**-0.0**), use the **-z** option on the **assign(1)** command.

IEEE_save=setting

(For Fortran) the ISO standard requires that any procedure which accesses the standard IEEE intrinsic modules via a "use" statement must save the floating point flags, halting mode, and rounding mode on entry; must restore the halting mode and rounding mode on exit; and must OR the saved flags with the current flags on exit. Setting this option **OFF** may improve

execution speed by skipping these steps.

recursive=*setting*

Invoke the language option control group to control recursion support. *setting* can be either **ON** or **OFF**. The default is **OFF**.

In either mode, the compiler supports a recursive, stack-based calling sequence. The difference lies in the optimization of statically allocated local variables, as described in the following paragraphs.

With **-LANG:recursive=ON**, the compiler assumes that a statically allocated local variable could be referenced or modified by a recursive procedure call. Therefore, such a variable must be stored into memory before making a call and reloaded afterwards.

With **-LANG:recursive=OFF**, the compiler can safely assume that a statically allocated local variable is not referenced or modified by a procedure call. This setting enables the compiler to optimize more aggressively.

rw_const=(ON|OFF)

Tell the compiler whether to treat a constant parameter in Fortran as read-only or read-write. If treated as read-write, the compiler has to generate extra code in passing these constant parameters so as to tolerate their being modified in the called function. The default is **OFF**, which is more efficient but will cause segmentation fault if the constant parameter is written into.

short_circuit_conditionals=(ON|OFF)

Handle **.AND.** and **.OR.** via short-circuiting, in which the second operand is not evaluated if unnecessary, even if it contains side effects. Default is **ON**. This flag is applicable only to Fortran, the flag has no effect on C/C++ programs.

-LIST: ...

The listing option flag controls information that gets written to a listing (**.lst**) file. The individual controls in this group are:

=(ON|OFF)

Enable or disable writing the listing file. The default is **ON** if any **-LIST:** group options are enabled. By default, the listing file contains a list of options enabled.

all_options[=(ON|OFF)]

Enable or disable listing of most supported options. The default is **OFF**.

notes[=(ON|OFF)]

If an assembly listing is generated (for example, on **-S**), various parts of the compiler (such as software pipelining) generate comments within the listing that describe what they have done. Specifying **OFF** suppresses these comments. The default is **ON**.

options[=(ON|OFF)]

Enable or disable listing of the options modified (directly in the command line, or indirectly as a side effect of other options). The default is **OFF**.

symbols[=(ON|OFF)]

Enable or disable listing of information about the symbols (variables) managed by the compiler.

-LNO: ...

Specify options and transformations performed on loop nests by the Loop Nest Optimizer (LNO). The **-LNO** options are enabled only if the optimization level of **-O3** or higher is in effect.

For information on the LNO options that are in effect during a compilation, use the **-LIST:all_options=ON** option.

-LNO:apo_use_feedback=(ON|OFF)

Effective only when specified with **-apo** under feedback-directed compilation, this flag tells the auto-parallelizer whether to use the feedback data of the loops in deciding whether each loop should be parallelized. When the compiler parallelizes a loop, it generates both a serial and a parallel version. If the trip count of the loop is small, it is not beneficial to use the parallel version during execution. When this flag is set to **ON** and the feedback data indicates that the loop has small trip count, the auto-parallelizer will not generate the parallel version, thus saving the runtime check needed to decide whether to execute the serial or parallel version of the loop. The default is **OFF**.

-LNO:build_scalar_reductions=(ON|OFF)

Build scalar reductions before any loop transformation analysis. Using this flag may enable further loop transformations involving reduction loops. The default is **OFF**. This flag is redundant when **-OPT:round-off=2** or greater is in effect.

-LNO:blocking=(ON|OFF)

Enable or disable the cache blocking transformation. The default is **ON**.

-LNO:blocking_size=N

This option specifies a block size that the compiler must use when performing any blocking. **N** must be a positive integer number that represents the number of iterations.

-LNO:fission=(0|1|2)

This option controls loop fission. The options can be one of the following:

0 = Disable loop fission (default)

1 = Perform normal fission as necessary

2 = Specify that fission be tried before fusion

Because **-LNO:fusion** is on by default, turning on fission without turning off fusion may result in their effects being nullified. Ordinarily, fusion is applied before fission. Specifying **-LNO:fission=2** will turn on fission and cause it to be applied before fusion.

-LNO:full_unroll,fu=N

Fully unroll loops with `trip_count <= N` inside LNO. **N** can be any integer between 0 and 100. The default value for **N** is 5. Setting this flag to 0 disables full unrolling of small trip count loops inside LNO.

-LNO:full_unroll_size=N

Fully unroll loops with unrolled loop size `<= N` inside LNO. **N** can be any integer between 0 and 10000. The conditions implied by the **full_unroll** option must also be satisfied for the loop to be fully unrolled. The default value for **N** is 2000.

-LNO:full_unroll_outer=(ON|OFF)

Control the full unrolling of loops with known trip count that do not contain a loop and are not contained in a loop. The conditions implied by both the **full_unroll** and the **full_unroll_size** options must be satisfied for the loop to be fully unrolled. The default is **OFF**.

-LNO:fusion=N

Perform loop fusion. **N** can be one of the following:

0 = Loop fusion is off

1 = Perform conservative loop fusion

2 = Perform aggressive loop fusion

The default is **1**.

-LNO:fusion_peeling_limit=N

This option sets the limit for the number of iterations allowed to be peeled in fusion, where $N \geq 0$. **N=5** by default.

-LNO:gather_scatter=N

This option enables gather-scatter optimizations. **N** can be one of the following:

- 0** = Disable all gather-scatter optimizations
- 1** = Perform gather-scatter optimizations in non-nested IF statements (default)
- 2** = Perform multi-level gather-scatter optimizations
- LNO:hoistif=(ON|OFF)**
This option enables or disables hoisting of IF statements inside inner loops to eliminate redundant loops. Default is **ON**.
- LNO:ignore_feedback=(ON|OFF)**
If the flag is **ON** then feedback information from the loop annotations will be ignored in LNO transformations. The default is **OFF**.
- LNO:ignore_pragmas=(ON|OFF)**
This option specifies that the command-line options override directives in the source file. Default is **OFF**.
- LNO:local_pad_size=N**
This option specifies the amount by which to pad local array dimensions. The compiler automatically (by default) chooses the amount of padding to improve cache behavior for local array accesses.
- LNO:minvariant,minvar=(ON|OFF)**
Enable or disable moving loop-invariant expressions out of loops. The default is **ON**.
- LNO:non_blocking_loads=(ON|OFF)**
(For C/C++ only) The option specifies whether the processor blocks on loads. If not set, the default of the current processor is used.
- LNO:oinvar=(ON|OFF)**
This option controls outer loop hoisting. Default is **ON**.
- LNO:opt=(0|1)**
This option controls the LNO optimization level. The options can be one of the following:
0 = Disable nearly all loop nest optimizations.
1 = Perform full loop nest transformations. This is the default.
- LNO:ou_prod_max=N**
This option indicates that the product of unrolling of the various outer loops in a given loop nest is not to exceed **N**, where **N** is a positive integer. The default is **16**.
- LNO:outer=(ON|OFF)**
This option enables or disables outer loop fusion. Default is **ON**.
- LNO:outer_unroll_max,ou_max=N**
The **Outer_unroll_max** option indicates that the compiler may unroll outer loops in a loop nest by as many as **N** per loop, but no more. The default is **5**.
- LNO:parallel_overhead=N**
Effective only when specified with **-apo**, the **parallel_overhead** option controls the auto-parallelizing compiler's estimate of the overhead (in processor cycles) incurred by invoking the parallel version of a loop. When the compiler parallelizes a loop, it generates both a serial and a parallel version. If the amount of work performed by the loop is small, it may not be beneficial to use the parallel version during execution. The set value of **parallel_overhead** is used in this determination during execution time when the number of processors and the iteration count of the loop are taken into account. The default value is **4096**. Because the optimal value varies across systems and programs, this option can be used for parallel performance tuning.
- LNO:prefetch=(0|1|2|3)**
This option specifies the level of prefetching.

0 = Prefetch disabled.

1 = Prefetch is done only for arrays that are always referenced in each iteration of a loop.

2 = Prefetch is done without the above restriction. This is the default.

3 = Most aggressive.

-LNO:prefetch_ahead=N

Prefetch **N** cache line(s) ahead. The default is **2**.

-LNO:prefetch_verbose=(ON|OFF)

-LNO:prefetch_verbose=ON prints verbose prefetch info to stdout. Default is **OFF**.

-LNO:processors=N

Tells the compiler to assume that the program compiled under **-apo** will be run on a system with the given number of processors. This helps in reducing the amount of computation during execution for determining whether to enter the parallel or serial versions of loops that are parallelized (see the **-LNO:parallel_ overhead** option). The default is **0**, which means unknown number of processors. The default value of **0** should be used if the program is intended to run in different systems with different number of processors. If the option is set to non-zero and the value is different from the number of processors, the parallelized code will not perform optimally.

-LNO:sclrze=(ON|OFF)

Turn **ON** or **OFF** the optimization that replaces an array by a scalar variable. The default is **ON**.

-LNO:simd=(0|1|2)

This flag controls inner loop vectorization which makes use of SIMD instructions provided by the native processor.

0 = Turn off the vectorizer.

1 = (Default) Vectorize only if the compiler can determine that there is no undesirable performance impact due to sub-optimal alignment. Vectorize only if vectorization does not introduce accuracy problems with floating-point operations.

2 = Vectorize without any constraints (most aggressive).

-LNO:simd_reduction=(ON|OFF)

This flag controls whether reduction loops will be vectorized. Default is **ON**.

-LNO:simd_verbose=(ON|OFF)

-LNO:simd_verbose=ON prints verbose vectorizer info to stdout. Default is **OFF**.

-LNO:svr_phase1=(ON|OFF)

This flag controls whether the scalar variable naming phase should be invoked before first phase of LNO. The default is **ON**.

-LNO:trip_count_assumed_when_unknown,trip_count=N

This flag is to provide an assumed loop trip-count if it is unknown at compile time. LNO uses this information for loop transformations and prefetch, etc. **N** can be any positive integer, and the default value is 1000.

-LNO:vintr=(0|1|2)

This flag controls loop vectorization to make use of vector intrinsic routines (Note: a vector intrinsic routine is called once to compute a math intrinsic for the entire vector). **-LNO:vintr=1** is the default. **-LNO:vintr=0** turns off the **vintr** optimization. Under **-LNO:vintr=2** the compiler will do aggressive optimization for all vector intrinsic routines. Note that **-LNO:vintr=2** could be unsafe in that some of these routines could have accuracy problems.

-LNO:vintr_verbose=(ON|OFF)

-LNO:vintr_verbose=ON prints verbose information to stdout on optimizing for vector intrinsic routines. Default is **OFF**. This flag will let you know which loops are vectorized to make use of vector

intrinsic routines.

Following are **LNO Transformation Options**. Loop transformation arguments allow control of cache blocking, loop unrolling, and loop interchange. They include the following options.

-LNO:interchange=(ON|OFF)

Disable the loop interchange transformation in the loop nest optimizer. Default is **ON**.

-LNO:unswitch=(ON|OFF)

Turn **ON** or **OFF** the optimization that performs a simple form of loop unswitching. The default is **ON**.

-LNO:unswitch_verbose=(ON|OFF)

-LNO:unswitch_verbose=ON prints verbose info to stdout on unswitching loops. Default is **OFF**.

-LNO:ou=N

This option indicates that all outer loops for which unrolling is legal should be unrolled by **N**, where **N** is a positive integer. The compiler unrolls loops by this amount or not at all.

-LNO:ou_deep=(ON|OFF)

This option specifies that for loops with 3-deep (or deeper) loop nests, the compiler should outer unroll the wind-down loops that result from outer unrolling loops further out. This results in large code size, but generates faster code (whenever wind-down loop execution costs are important). Default is **ON**.

-LNO:ou_further=N

This option specifies whether or not the compiler performs outer loop unrolling on wind-down loops. **N** must be specified and be an integer.

Additional unrolling can be disabled by specifying **-LNO:ou_further=999999**. Unrolling is enabled as much as is sensible by specifying **-LNO:ou_further=3**.

-LNO:ou_max=N

This option enables the compiler to unroll as many as **N** copies per loop, but no more.

-LNO:pwr2=(ON|OFF)

(For C/C++ only) This option specifies whether to ignore the leading dimension (set this to **OFF** to ignore).

Following are **LNO Target Cache Memory Options**. These arguments allow you to describe the target cache memory system. In the following arguments, the numbering starts with the cache level closest to the processor and works outward.

-LNO:assoc1=N, assoc2=N, assoc3=N, assoc4=N

This option specifies the cache set associativity. For a fully associative cache, such as main memory, **N** should be set to any sufficiently large number, such as 128. Specify a positive integer for **N**; specifying **N=0** indicates there is no cache at that level.

-LNO:cmp1=N, cmp2=N, cmp3=N, cmp4=N, dmp1=N, dmp2=N, dmp3=N, dmp4=N

This option specifies, in processor cycles, the time for a clean miss (**cmpx=**) or a dirty miss (**dmpx=**) to the next outer level of the memory hierarchy. This number is approximate because it depends on a clean or dirty line, read or write miss, etc. Specify a positive integer for **N**; specifying **N=0** indicates there is no cache at that level.

-LNO:cs1=N, cs2=N, cs3=N, cs4=N

This option specifies the cache size. **N** can be 0 or a positive integer followed by one of the following letters: k, K, m, or M. These letters specify the cache size in Kbytes or Mbytes. Specifying **0** indicates there is no cache at that level.

cs1 is the primary cache, **cs2** refers to the secondary cache, **cs3** refers to memory, and **cs4** is the disk. Default cache size for each type of cache depends on your system. Use **-LIST:all_options=ON** to see the default cache sizes used during compilation.

-LNO:is_mem1=(ON|OFF), is_mem2=(ON|OFF), is_mem3=(ON|OFF), is_mem4=(ON|OFF)

This option specifies that certain memory hierarchies should be modeled as memory not cache. Default is **OFF** for each option.

Blocking can be attempted for this memory level, and blocking appropriate for memory, rather than cache, is applied. No prefetching is performed, and any prefetching options are ignored. If **-OPT:is_memx=(ON|OFF)** is specified, the corresponding **assocx=N** specification is ignored, any **cmpx=N** and **dmpx=N** options on the command line are ignored.

-LNO:ls1=N, ls2=N, ls3=N, ls4=N

This option specifies the line size in bytes. This is the number of bytes, specified in the form of a positive integer number (N), that are moved from the memory hierarchy level further out to this level on a miss. Specifying **N=0** indicates there is no cache at that level.

Following are **LNO TLB Options**. These arguments control the TLB, a cache for the page table, assumed to be fully associative. The TLB control arguments are the following.

-LNO:ps1=N, ps2=N, ps3=N, ps4=N

This option specifies the number of bytes in a page, with N as positive integer. The default for N depends on your system hardware.

-LNO:tlb1=N, tlb2=N, tlb3=N, tlb4=N

This option specifies the number of entries in the TLB for this cache level, with N as a positive integer. The default for N depends on your system hardware.

-LNO:tlbcmp1=N, tlbcmp2=N, tlbcmp3=N, tlbcmp4=N, tlbdmp1=N, tlbdmp2=N, tlbdmp3=N, tlbdmp4=N

This option specifies the number of processor cycles it takes to service a clean TLB miss (the **tlbcmpx=** options) or a dirty TLB miss (the **tlbdmpx=** options), with N as a positive integer. The default for N depends on your system hardware.

Following are **LNO Prefetch Options**. These arguments control the prefetch operation.

-LNO:assume_unknown_trip_count={0,1000}

This flag is no longer supported. It has been promoted to **-LNO:trip_count_assumed_when_unknown**

-LNO:pf1=(ON|OFF), pf2=(ON|OFF), pf3=(ON|OFF), pf4=(ON|OFF)

This options selectively disables or enables prefetching for cache level x, for **pfx=(ON|OFF)**

-LNO:prefetch=(0|1|2|3)

This option specifies the levels of prefetching. The options can be one of the following:

0 = Prefetch disabled.

1 = Prefetch is done only for arrays that are always referenced in each iteration of a loop.

2 = Prefetch is done without the above restriction. This is the default.

3 = Most aggressive.

-LNO:prefetch_ahead=N

This option prefetches the specified number of cache lines ahead of the reference. Specify a positive integer for N; default is **2**.

-LNO:prefetch_manual=(ON|OFF)

This option specifies whether manual prefetches (through directives) should be respected or ignored.

prefetch_manual=OFF ignores directives for prefetches.

prefetch_manual=ON respects directives for prefetches. This is the default.

-M Run `cpp` and print list of make dependencies.

-m32 Compile for 32-bit ABI, also known as x86 or IA32. See `-m64` for defaults.

-m3dnow

Enable use of 3DNow instructions. The default is **OFF**.

- m64** Compile for 64-bit ABI, also known as AMD64, x86_64, or IA32e. On a 32-bit host, the default is 32-bit ABI. On a 64-bit host, the default is 64-bit ABI if the target platform (`-march/-mcpu/-mtune`) is 64-bit; otherwise the default is 32-bit.
- macro-expand**
Enable macro expansion in preprocessed Fortran source files throughout each file. Without this option specified, macro expansion is limited to preprocessor `#` directives in files processed by the Fortran preprocessor. When this option is specified, macro expansion occurs throughout the source file.
- march=<cpu-type>**
Compiler will optimize code for the selected cpu type: **opteron**, **athlon**, **athlon64**, **athlon64fx**, **em64t**, **pentium4**, **xeon**, **core**, **anyx86**, **auto**. **auto** means to optimize for the platform that the compiler is running on, which the compiler determines by reading `/proc/cpuinfo`. **anyx86** means a generic x86 processor. Under 32-bit ABI, anyx86 is a processor without SSE2/SSE3/3DNow! support; under 64-bit ABI it is a processor with SSE2 but without SSE3/3DNow!. **Core** refers to the Intel Core Microarchitecture, used by 64-bit CPUs such as Woodcrest. The default is **auto**.
- mcmode=(small|medium)**
Select the code size model to use when generating offsets within object files. Most programs will work with **-mcmode=small** (using 32-bit pointers), but some need **-mcmode=medium** (using 32-bit pointers for code and 64-bit pointers for data).
- mcpu=<cpu-type>**
Behaves like **-march**. See **-march**.
- MD** Write dependencies to `.d` output file
- MDtarget**
Use the following as the target for Make dependencies.
- MDupdate**
Update the following file with Make dependencies.
- MF** Write dependencies to specified output file.
- MG** With **-M** or **-MM**, treat missing header files as generated files.
- MM** Output user dependencies of source file.
- MMD** Write user dependencies to `.d` output file.
- mno-sse**
Disable the use of SSE2/SSE3 instructions. SSE2 cannot be disabled under **-m64** and will result in a warning.
- mno-sse2**
Disable the use of SSE2/SSE3 instructions. SSE2 cannot be disabled under **-m64** and will result in a warning.
- mno-sse3**
Disable the use of SSE3 instructions.
- module dir**
Create the `.mod` file corresponding to a `"module"` statement in the directory `dir` instead of the current working directory. Also, when searching for modules named in `"use"` statements, examine the directory `dir` before the directories established by **-I**`dir` options.
- mp** Interpret OpenMP directives to explicitly parallelize regions of code for execution by multiple threads on a multi-processor system. Most OpenMP 2.0 directives are supported by **pathf95**, **pathcc** and **pathCC**. See the *QLogic PathScale Compiler Suite User Guide* for more information on these directives.
- MP** With **-M** or **-MM**, add phony targets for each dependency.

- MQ** Same as **-MT**, but quote characters that are special to Make.
- msse2** Enable use of SSE2 instructions. This is the default under both **-m64** and **-m32**.
- msse3** Enable use of SSE3 instructions. Default is **ON** under **-march=em64t** and **-march=core**. Otherwise, it is **OFF** by default.
- mtune=<cpu-type>**
Behaves like **-march**. See **-march**.
- MT** Change the target of the generated dependency rules.
- mx87-precision=(32|64|80)**
Specify the precision of x87 floating-point calculations. The default is **80**-bits.
- nobool** Do not allow boolean keywords.
- nocpp** (For **Fortran** only) Disable the source preprocessor.
See the **-cpp**, **-E**, and **-ftpp** options for more information on controlling preprocessing.
- nodefaultlibs**
Do not use standard system libraries when linking.
- noexpopt**
Do not optimize exponentiation operations.
- noextend-source**
Restrict Fortran source code lines to columns 1 through 72.
See the **-coln** and **-extend-source** options for more information on controlling line length.
- nog77mangle**
The PathScale Fortran compiler modifies Fortran symbol names by appending an underscore, so a name like "foo" in a source file becomes "foo_" in an object file.
However, if a name in a Fortran source file contains an underscore, the compiler appends a second underscore in the object file, so "foo_bar" becomes "foo_bar__", and "baz_" becomes "baz___".
The **-nog77mangle** option suppresses the addition of this second underscore.
- no-gcc** (For C/C++ only) **-no-gcc** turns off the **__GNUCC__** and other predefined preprocessor macros.
- noinline**
Suppress expansion of inline functions. When this option is specified, copies of inline functions are emitted as static functions in each compilation unit where they are called. If you are using IPA, **-IPA:inline=OFF** must be specified to suppress inlining.
- no-pathcc**
-no-pathcc turns off the **__PATHSCALE__** and other predefined preprocessor macros.
- nostartfiles**
Do not use standard system startup files when linking.
- nostdinc**
Direct the system to skip the standard directory, **/usr/include**, when searching for **#include** files and files named on **INCLUDE** statements.
- nostdinc++**
Do not search for header files in the standard directories specific to C++.
- nostdlib**
No predefined libraries or startfiles.
- o outfile**
When this option is used in conjunction with the **-c** option and a single C source file, a relocatable object file named *outfile* is produced. When specified with the **-S** option, the **-o** option is ignored. If **-o** and **-c** are not specified, a file named **a.out** is produced. If specified, writes the executable file to *out_file* rather

than to **a.out**.

-O(0|1|2|3|s)

Specify the basic level of optimization desired. The options can be one of the following:

- 0** Turn off all optimizations.
- 1** Turn on local optimizations that can be done quickly.
- 2** Turn on extensive optimization. This is the default. The optimizations at this level are generally conservative, in the sense that they are virtually always beneficial, provide improvements commensurate to the compile time spent to achieve them, and avoid changes which affect such things as floating point accuracy.
- 3** Turn on aggressive optimization. The optimizations at this level are distinguished from **-O2** by their aggressiveness, generally seeking highest-quality generated code even if it requires extensive compile time. They may include optimizations that are generally beneficial but may hurt performance.

This includes but is not limited to turning on the Loop Nest Optimizer, **-LNO:opt=1**, and setting **-OPT:ro=1:IEEE_arith=2:Olimit=9000:reorg_common=ON**.

s Specify that code size is to be given priority in tradeoffs with execution time.

If no value is specified, **2** is assumed.

-objectlist

Read the following file to get a list of files to be linked.

-Ofast Equivalent to **-O3 -ipa -OPT:Ofast -fno-math-errno -ffast-math**. Use optimizations selected to maximize performance. Although the optimizations are generally safe, they may affect floating point accuracy due to rearrangement of computations.

NOTE: **-Ofast** enables **-ipa** (inter-procedural analysis), which places limitations on how libraries and **.o** files are built.

-openmp

Interpret OpenMP directives to explicitly parallelize regions of code for execution by multiple threads on a multi-processor system. Most OpenMP 2.0 directives are supported by **pathf95**, **pathcc** and **pathCC**. See the *QLogic PathScale Compiler Suite User Guide* for more information on these directives.

-OPT:...

This option group controls miscellaneous optimizations. These options override defaults based on the main optimization level.

-OPT:alias=<name>

Specify the pointer aliasing model to be used. By specifying one or more of the following for *<name>*, the compiler is able to make assumptions throughout the compilation:

- typed** Assume that the code adheres to the ANSI/ISO C standard which states that two pointers of different types cannot point to the same location in memory. This is **ON** by default when **-OPT:Ofast** is specified.
- restrict** Specify that distinct pointers are assumed to point to distinct, non-overlapping objects. This is **OFF** by default.
- disjoint** Specify that any two pointer expressions are assumed to point to distinct, non-overlapping objects. This is **OFF** by default.

-OPT:align_unsafe=(ON|OFF)

Instruct the vectorizer (invoked at **-O3**) to aggressively perform vectorization by assuming that array parameters are aligned at 128-bit boundaries. The vectorizer will then generate 128-bit aligned load and store instructions, which are faster than their unaligned counterparts. If the assumption is incorrect, the aligned memory accesses will result in run-time segmentation faults. The default is **OFF**.

-OPT:asm_memory=(ON|OFF)

A debugging option to be used when debugging suspected buggy inline assembly. If **ON**, the compiler assumes each asm has "memory" specified even if it is not there. The default is **OFF**.

-OPT:bb=N

This specifies the maximum number of instructions a basic block (straight line sequence of instructions with no control flow) can contain in the code generator's program representation. Increasing this value can improve the quality of optimizations that are applied at the basic block level, but can increase compilation time in programs that exhibit such large basic blocks. The default is 1300. If compilation time is an issue, use a smaller value.

-OPT:cis=(ON|OFF)

Convert SIN/COS pairs using the same argument to a single call calculating both values at once. The default is **ON**.

-OPT:div_split=(ON|OFF)

Enable or disable changing x/y into x*(recip(y)). This is **OFF** by default, but enabled by **-OPT:Ofast** or **-OPT:IEEE_arithmetic=3**. This transformation generates fairly accurate code.

-OPT:early_mp=(ON|OFF)

This flag has any effect only under **-mp** compilation. It controls whether the transformation of code to run under multiple threads should take place before or after the loop nest optimization (LNO) phase in the compilation process. The default is **OFF**, when the transformation occurs after LNO. Some OpenMP programs can yield better performance by enabling **-OPT:early_mp** because LNO can sometimes generate more appropriate loop transformation when working on the multi-threaded forms of the loops. If **-apo** is specified, the transformation of code to run under multiple threads can only take place after the LNO phase, in which case this flag is ignored.

-OPT:early_intrinsics=(ON|OFF)

When **ON**, this option causes calls to intrinsics to be expanded to inline code early in the backend compilation. This may enable more vectorization opportunities if vector forms of the expanded operations exist. Default is **OFF**.

-OPT:fast_bit_intrinsics=(ON|OFF)

Setting this to **ON** will turn off the check for the bit count being within range for Fortran intrinsics (like BTEST and ISHFT). The default setting is **OFF**.

-OPT:fast_complex=(ON|OFF)

Setting **fast_complex=ON** enables fast calculations for values declared to be of the type *complex*. When this is set to **ON**, complex absolute value (norm) and complex division use fast algorithms that overflow for an operand (the divisor, in the case of division) that has an absolute value that is larger than the square root of the largest representable floating-point number. This would also apply to an underflow for a value that is smaller than the square root of the smallest representable floating point number. **OFF** is the default. **fast_complex=ON** is enabled if **-OPT:roundoff=3** is in effect.

-OPT:fast_exp=(ON|OFF)

This option enables optimization of exponentiation by replacing the runtime call for exponentiation by multiplication and/or square root operations for certain compile-time constant exponents (integers and halves). This can produce differently rounded results than those from the runtime function. **fast_exp** is **OFF** unless **-O3** or **-Ofast** are specified, or **-OPT:roundoff=1** is in effect.

-OPT:fast_io=(ON|OFF)

(For C/C++ only) This option enables inlining of printf(), fprintf(), sprintf(), scanf(), fscanf(), sscanf(), and printw(). **-OPT:fast_io** is only in effect when the candidates for inlining are marked as intrinsic to the stdio.h and curses.h files. Default is **OFF**.

-OPT:fast_math=(ON|OFF)

Setting this to **ON** will tell the compiler to use the fast math functions tuned for the processor. The affected math functions include log, exp, sin, cos, sincos, expf and pow. The default setting is **OFF**. It is turned

on automatically when **-OPT:roundoff** is at 2 or above.

-OPT:fast_nint=(ON|OFF)

This option uses hardware features to implement NINT and ANINT (both single- and double-precision versions). Default is **OFF** but **fast_nint=ON** is enabled by default if **-OPT:roundoff=3** is in effect.

-OPT:fast_sqrt=(ON|OFF)

This option calculates square roots using the identity $\text{sqrt}(x)=x*\text{rsqrt}(x)$, where **rsqrt** is the reciprocal square root operation. This transformation generates fairly accurate code. Default is **OFF**. (Note that in order for **-OPT:fast_sqrt=ON** to take effect, **-OPT:fast_exp** must be **ON** which tells the compiler to emit inlined instructions instead of calling the library **pow** function. Also note that **-OPT:fast_sqrt** is independent of **-OPT:rsqrt**, which transforms $1/\text{sqrt}(x)$ to **rsqrt**(x). Unlike **-OPT:rsqrt**, the compiler does not generate extra code to refine the **rsqrt** result for **-OPT:fast_sqrt**.)

-OPT:fast_stdlib=(ON|OFF)

This option controls the generation of calls to faster versions of some standard library functions. Default is **ON**.

-OPT:fast_trunc=(ON|OFF)

This option inlines the NINT, ANINT, and AMOD Fortran intrinsics, both single- and double-precision versions. Default is **OFF**. **fast_trunc** is enabled automatically if **-OPT:roundoff=1** or greater is in effect.

-OPT:fold_reassociate=(ON|OFF)

This option allows optimizations involving reassociation of floating point quantities. Default is **OFF**. **fold_reassociate=ON** is enabled automatically when **-OPT:roundoff=2** or greater is in effect.

-OPT:fold_unsafe_relops=(ON|OFF)

This option folds relational operators in the presence of possible integer overflow. The default is **ON** for **-O3** and **OFF** otherwise.

-OPT:fold_unsigned_relops=(ON|OFF)

This option folds unsigned relational operators in the presence of possible integer overflow. Default is **OFF**.

-OPT:goto=(ON|OFF)

Disable or enable the conversion of GOTOs into higher-level structures like FOR loops. The default is **ON** for **-O2** or higher.

-OPT:IEEE_arithmetic,IEEE_arith=(1|2|3)

Specify the level of conformance to IEEE 754 floating pointing roundoff/overflow behavior. Note that **-OPT:IEEE_a** is a valid abbreviation for this flag. The options can be one of the following:

- 1 Adhere to IEEE accuracy. This is the default when optimization levels **-O0**, **-O1** and **-O2** are in effect.
- 2 May produce inexact result not conforming to IEEE 754. This is the default when **-O3** is in effect.
- 3 All mathematically valid transformations are allowed.

-OPT:IEEE_NaN_Inf=(ON|OFF)

-OPT:IEEE_NaN_inf=ON forces all operations that might have IEEE-754 NaN or infinity operands to yield results that conform to ANSI/IEEE 754-1985, the IEEE Standard for Binary Floating-point Arithmetic, which describes a standard for NaN and inf operands. Default is **ON**.

-OPT:IEEE_NaN_inf=OFF produces non-IEEE results for various operations. For example, **x=x** is treated as **TRUE** without executing a test and **x/x** is simplified to **1** without dividing. **OFF** can enable many common optimizations that can help performance.

-OPT:inline_intrinsics=(ON|OFF)

When **OFF**, this option turns all Fortran intrinsics that have a library function into a call to that function. Default is **ON**.

-OPT:malloc_algorithm=(0|1) or -OPT:malloc_alg=(0|1)

Select an alternate malloc algorithm which may improve speed. The compiler adds setup code in the C/C++/Fortran "main" function to enable the chosen algorithm. The default is **0**.

-OPT:Ofast

Use optimizations selected to maximize performance. Although the optimizations are generally safe, they may affect floating point accuracy due to rearrangement of computations. This effectively turns on the following optimizations: **-OPT:ro=2:Olimit=0:div_split=ON:alias=typed**.

-OPT:Olimit=N

Disable optimization when size of program unit is > N. When N is 0, program unit size is ignored and optimization process will not be disabled due to compile time limit. The default is **0** when **-OPT:Ofast** is specified, **9000** when **-O3** is specified; otherwise the default is **6000**.

-OPT:pad_common=(ON|OFF)

This option reorganizes common blocks to improve the cache behavior of accesses to members of the common block. This may involve adding padding between members and/or breaking a common block into a collection of blocks. Default is **OFF**.

This option should not be used unless the common block definitions (including EQUIVALENCE) are consistent among all sources making up a program. In addition, **pad_common=ON** should not be specified if common blocks are initialized with DATA statements. If specified, **pad_common=ON** must be used for all of the source files in the program.

-OPT:recip=(ON|OFF)

This option specifies that faster, but potentially less accurate, reciprocal operations should be performed. Default is **OFF**.

-OPT:reorg_common=(ON|OFF)

This option reorganizes common blocks to improve the cache behavior of accesses to members of the common block. The reorganization is done only if the compiler detects that it is safe to do so.

reorg_common=ON is enabled when **-O3** is in effect and when all of the files that reference the common block are compiled at **-O3**.

reorg_common=OFF is set when the file that contains the common block is compiled at **-O2** or below.

-OPT:roundoff=(0|1|2|3) or -OPT:ro=(0|1|2|3)

Specify the level of acceptable departure from source language floating-point, round-off, and overflow semantics. The options can be one of the following:

0 = Inhibit optimizations that might affect the floating-point

behavior. This is the default when optimization levels **-O0**, **-O1**, and **-O2** are in effect.

1 = Allow simple transformations that might cause limited

round-off or overflow differences. Compounding such transformations could have more extensive effects. This is the default when **-O3** is in effect.

2 = Allow more extensive transformations, such as the

reordering of reduction loops. This is the default level when **-OPT:Ofast** is specified.

3 = Enable any mathematically valid transformation.

-OPT:rsqrt=(0|1|2)

This option calculates reciprocal square roots using the rsqrt machine instruction. rsqrt is faster but potentially less accurate than the regular square root operation. 0 means not to use rsqrt. 1 means to use rsqrt followed by instructions to refine the result. 2 means to use rsqrt by itself. Default is **1** when **-OPT:roundoff=2** or greater, else the default is **0**.

-OPT:space=(ON|OFF) When **ON**, this option specifies that

code size is to be given priority in tradeoffs with execution time in optimization choices. Default is **OFF**. This can be turned on either directly or by compiling with **-Os**.

- OPT:speculate=(ON|OFF)** When **ON**, this option makes the compiler convert short-circuiting conditionals to their equivalent non-short-circuited forms whenever possible. This eliminates branches at the expense of more computations. Default is **OFF**.
- OPT:transform_to_memlib=(ON|OFF)**
When **ON**, this option enables transformation of loop constructs to calls to **memcpy** or **memset**. Default is **ON**.
- OPT:treeheight=(ON|OFF)**
The value **ON** enables re-association in expressions to reduce the expressions' tree height. The default is **OFF**.
- OPT:unroll_analysis=(ON|OFF)**
The default value of **ON** lets the compiler analyze the content of the loop to determine the best unrolling parameters, instead of strictly adhering to the **-OPT:unroll_times_max** and **-OPT:unroll_size** parameters.
-OPT:unroll_analysis=ON can have the negative effect of unrolling loops less than the upper limit dictated by the **-OPT:unroll_times_max** and **-OPT:unroll_size** specifications.
- OPT:unroll_times_max=N**
Unroll inner loops by a maximum of **N**. The default is **4**.
- OPT:unroll_size=N**
Set the ceiling of maximum number of instructions for an unrolled inner loop. If **N=0**, the ceiling is disregarded. The default is **40**.
- OPT:wrap_around_unsafe_opt=(ON|OFF)**
-OPT:wrap_around_unsafe_opt=OFF disables both the induction variable replacement and linear function test replacement optimizations. By default these optimizations are enabled at **-O3**. This option is disabled by default at **-O0**.
Setting **-OPT:wrap_around_unsafe_opt** to **OFF** can degrade performance. It is provided as a diagnostic tool.
- P** When used with **-E**, the source preprocessor will not generate **#** lines in the output.
- pad-char-literals**
(For **Fortran** only) Blank pad all character literal constants that are shorter than the size of the default integer type and that are passed as actual arguments. The padding extends the length to the size of the default integer type.
- pathcc** Define **__PATHCC__** and other macros.
- pedantic-errors**
Issue warnings needed by strict compliance to ANSI C.
- pg** Generate extra code to profile information suitable for the analysis program **pathprof(1)**. You must use this option when compiling the source files you want data about, and you must also use it when linking. This option turns on application level profiling but not library level profiling (see also **-profile**). See the **gcc** man pages for more information.
- profile** Generate extra code to profile information suitable for the analysis program **pathprof(1)**. You must use this option when compiling the source files you want data about, and you must also use it when linking. This option turns on application level and library level profiling (see also **-pg**).
- r** Produce a relocatable **.o** and stop.
- rreal_spec**
(For **Fortran** only) Specify the default kind specification for real values.

| Option | Kind value |
|------------|--|
| -r4 | Use REAL(KIND=4) and COMPLEX(KIND=4) for real and complex variables, respectively (the default). |

- r8** Use **REAL(KIND=8)** and **COMPLEX(KIND=8)** for real and complex variables, respectively.
- S** Generate an assembly file, *file.s*, rather than an object file (*file.o*).
- shared** DSO–shared PIC code.
- shared–libgcc**
Force the use of the shared libgcc library.
- show** Print the passes as they execute with their arguments and their input and output files.
- show-defaults**
Show the processor target settings and the default options in the **compiler.defaults(5)** file. For C/C++, also shows the GNU GCC version compatibility.
- show0** Show what phases would be called, but don't invoke anything.
- showt** Show time taken by each phase.
- static** Suppress dynamic linking at runtime for shared libraries; use static linking instead.
- static–data**
Statically allocate all local variables. Statically allocated local variables are initialized to zero and exist for the life of the program. This option can be useful when porting programs from older systems in which all variables are statically allocated.

When compiling with the **-static–data** option, global data is allocated as part of the compiled object (*file.o*) file. The total size of any *file.o* cannot exceed 2 GB, but the total size of a program loaded from multiple *.o* files can exceed 2 GB. An individual common block cannot exceed 2 GB, but you can declare multiple common blocks each having that size.

If a parallel loop in a multi-processed program calls an external routine, that external routine cannot be compiled with the **-static–data** option. You can mix static and multi-processed object files in the same executable, but a static routine cannot be called from within a parallel region.
- static–libgcc**
Force the use of the static libgcc library.
- std=c++98**
-std option for g++.
- std=c89**
-std option for gcc/g++.
- std=c99**
-std option for gcc/g++.
- std=c9x**
-std option for gcc/g++.
- std=gnu++98**
-std option for g++.
- std=gnu89**
-std option for gcc/g++.
- std=gnu99**
-std option for gcc/g++.
- std=gnu9x**
-std option for gcc/g++.
- std=iso9899:1990**
-std option for gcc/g++.

- std=iso9899:199409**
-std option for gcc/g++.
- std=iso9899:1999**
-std option for gcc/g++.
- std=iso9899:199x**
-std option for gcc/g++.
- stdinc** Predefined include search path list.
- subverbose**
Produce diagnostic output about the subscription management for the compiler.
- TENV: ...**
This option specifies the target environment option group. These options control the target environment assumed and/or produced by the compiler.
- TENV:frame_pointer=(ON|OFF)**
Default is **ON** for C++ and **OFF** otherwise. Local variables in the function stack frame are addressed via the frame pointer register. Ordinarily, the compiler will replace this use of frame pointer by addressing local variables via the stack pointer when it determines that the stack pointer is fixed throughout the function invocation. This frees up the frame pointer for other purposes. Turning this flag on forces the compiler to use the frame pointer to address local variables. This flag defaults to **ON** for C++ because the exception handling mechanism relies on the frame pointer register being used to address local variables. This flag can be turned **OFF** for C++ for programs that do not throw exceptions.
- TENV:X=(0..4)**
Specify the level of enabled exceptions that will be assumed for purposes of performing speculative code motion (default is level 1 at all optimization levels). In general, an instruction will not be speculated (i.e. moved above a branch by the optimizer) unless any exceptions it might cause are disabled by this option.
 - Level **0** - No speculative code motion may be performed.
 - Level **1** - Safe speculative code motion may be performed, with IEEE-754 underflow and inexact exceptions disabled.
 - Level **2** - All IEEE-754 exceptions are disabled except divide by zero.
 - Level **3** - All IEEE-754 exceptions are disabled including divide by zero.
 - Level **4** - Memory exceptions may be disabled or ignored.
- TENV:simd_imask=(ON|OFF)**
Default is **ON**. Turning it **OFF** unmask SIMD floating-point invalid-operation exception.
- TENV:simd_dmask=(ON|OFF)**
Default is **ON**. Turning it **OFF** unmask SIMD floating-point denormalized-operand exception.
- TENV:simd_zmask=(ON|OFF)**
Default is **ON**. Turning it **OFF** unmask SIMD floating-point zero-divide exception.
- TENV:simd_omask=(ON|OFF)**
Default is **ON**. Turning it **OFF** unmask SIMD floating-point overflow exception.
- TENV:simd_umask=(ON|OFF)**
Default is **ON**. Turning it **OFF** unmask SIMD floating-point underflow exception.
- TENV:simd_pmask=(ON|OFF)**
Default is **ON**. Turning it **OFF** unmask SIMD floating-point precision exception.
- traditional**
Attempt to support traditional K&R style C.
- trapuv** Trap uninitialized variables. Initialize variables to the value NaN, which helps your program crash if it uses uninitialized variables. Affects local scalar and array variables and memory returned by alloca().

Does not affect the behavior of globals, malloc(ed) memory, or Fortran common data.

- U *name***
Remove any initial definition of *name*.
- Uvar** Undefine a variable for the source preprocessor. See the **-Dvar** option for information on defining variables.
- uvar** Make the default type of a variable undefined, rather than using default Fortran 90 rules.
- v** Print (on standard error output) the commands executed to run the stages of compilation. Also print the version number of the compiler driver program and of the preprocessor and the compiler proper.
- version**
Write compiler release version information to **stdout**. No input file needs to be specified when this option is used.
- Wc, arg1[,arg2...]**

Pass the argument(s) *argi* to the compiler pass *c* where *c* is one of [**pfibal**]. The **c** selects the compiler pass according to the following table:

| Character | Name |
|-----------|--------------|
| p | preprocessor |
| f | front-end |
| i | inliner |
| b | backend |
| a | assembler |
| l | loader |

Sets of these phase names can be used to select any combination of phases. For example, **-Wba,-o,foo** passes the option **-o foo** to the **b** and **a** phases.

- Wall** Enable most warning messages.
- WB,:** **-WB,<arg>** passes *<arg>* to the backend via ipacom.
- Wdeclaration-after-statement**
(For C/C++ only) Warn about declarations after statements (pre-C99).
- Werror-implicit-function-declaration**
(For C/C++ only) Give an error when a function is used before being declared.
- W[no-]aggregate-return**
(For C/C++ only) **-Waggregate-return** warns about returning structures, unions or arrays. **-Wno-aggregate-return** will not warn about returning structures, unions, or arrays.
- W[no-]bad-function-cast**
-Wbad-function-cast attempts to support writable-strings K&R style C. **-Wno-bad-function-cast** tells the compiler not to warn when a function call is cast to a non-matching type.
- W[no-]cast-align**
(For C/C++ only) **-Wcast-align** warns about pointer casts that increase alignment. **-Wno-cast-align** instructs the compiler not warn about pointer casts that increase alignment.
- Wno-cast-qual**
(For C/C++ only) **-Wcast-qual** warns about casts that discard qualifiers. **-Wno-cast-qual** tells the compiler not to warn about casts that discard qualifiers.
- W[no-]char-subscripts**
(For C/C++ only) **-Wchar-subscripts** warns about subscripts whose type is 'char'. The **-Wno-char-subscripts** option tells the compiler not warn about subscripts whose type is 'char'.

- W[no-]comment**
(For C/C++ only) **-Wcomment** warns if nested comments are detected. **-Wno-comment** tells the compiler not to warn if nested comments are detected.
- W[no-]conversion**
(For C/C++ only) **-Wconversion** warns about possibly confusing type conversions. **-Wno-conversion** tells the compiler not to warn about possibly confusing type conversions.
- W[no-]deprecated**
-Wdeprecated will announce deprecation of compiler features. **-Wno-deprecated** tells the compiler not to announce deprecation of compiler features.
- Wno-deprecated-declarations**
Do not warn about deprecated declarations in code.
- W[no-]disabled-optimization**
-Wdisabled-optimization warns if a requested optimization pass is disabled. **-Wno-disabled-optimization** tells the compiler not to warn if a requested optimization pass is disabled.
- W[no-]div-by-zero**
-Wdiv-by-zero warns about compile-time integer division by zero. **-Wno-div-by-zero** suppresses warnings about compile-time integer division by zero.
- W[no-]endif-labels**
-Wendif-labels warns if #if or #endif is followed by text. **-Wno-endif-labels** tells the compiler not to warn if #if or #endif is followed by text.
- W[no-]error**
-Werror makes all warnings into errors. **-Wno-error** tells the compiler not to make all warnings into errors.
- W[no-]float-equal**
-Wfloat-equal warns if floating point values are compared for equality. **-Wno-float-equal** tells the compiler not to warn if floating point values are compared for equality.
- W[no-]format**
(For C/C++ only) **-Wformat** warns about printf format anomalies. **-Wno-format** tells the compiler not to warn about printf format anomalies.
- Wno-format-extra-args**
(For C/C++ only) Do not warn about extra arguments to printf-like functions.
- W[no-]format-nonliteral**
(For C/C++ only) With the **-Wformat-nonliteral** option, and if **-Wformat**, warn if format string is not a string literal. For **-Wno-format-nonliteral** do not warn if format string is not a string literal.
- W[no-]format-security**
(For C/C++ only) For **-Wformat-security**, if **-Wformat**, warn on potentially insecure format functions. **-Wfno-format-security**, do not warn on potentially insecure format functions.
- Wno-format-y2k**
(For C/C++ only) Do not warn about 'strftime' formats that yield two-digit years.
- W[no-]id-clash**
(For C/C++ only) **-Wid-clash** warns if two identifiers have the same first <num> chars. **-Wno-id-clash** tells the compiler not to warn if two identifiers have the same first <num> chars.
- W[no-]implicit**
(For C/C++ only) **-Wimplicit** warns about implicit declarations of functions or variables. **-Wno-implicit** tells the compiler not to warn about implicit declarations of functions or variables.
- W[no-]implicit-function-declaration**
(For C/C++ only) **-Wimplicit-function-declaration** warns when a function is used before being declared. **-Wno-implicit-function-declaration** tells the compiler not to warn when a function is used before being

declared.

-W[no-]implicit-int

(For C/C++ only) **-Wimplicit-int** warns when a declaration does not specify a type. **-Wno-implicit-int** tells the compiler not to warn when a declaration does not specify a type.

-W[no-]import

-Wimport warns about the use of the `#import` directive. **-Wno-import** tells the compiler not to warn about the use of the `#import` directive.

-W[no-]inline

(For C/C++ only) **-Winline** warns if a function declared as inline cannot be inlined. **-Wno-inline** tells the compiler not to warn if a function declared as inline cannot be inlined.

-W[no-]larger-than-<number>

-Wlarger-than- warns if an object is larger than `<number>` bytes. **-Wno-larger-than-** tells the compiler not to warn if an object is larger than `<number>` bytes.

-Wno-long-long

(For C/C++ only) **-Wlong-long** warns if the long long type is used. **-Wno-long-long** tells the compiler not to warn if the long long type is used.

-W[no-]main

(For C/C++ only) **-Wmain** warns about suspicious declarations of main. **-Wno-main** tells the compiler not to warn about suspicious declarations of main.

-W[no-]missing-braces

(For C/C++ only) **-Wmissing-braces** warns about possibly missing braces around initializers. **-Wno-missing-braces** tells the compiler not to warn about possibly missing braces around initializers.

-W[no-]missing-declarations

(For C/C++ only) **-Wmissing-declarations** warns about global funcs without previous declarations. **-Wno-missing-declarations** tells the compiler not to warn about global funcs without previous declarations.

-W[no-]missing-format-attribute

(For C/C++ only) For the **-Wmissing-format-attribute** option, if **-Wformat** is used, warn on candidates for 'format' attributes. For **-Wno-missing-format-attribute** do not warn on candidates for 'format' attributes.

-W[no-]missing-noreturn

(For C/C++ only) **-Wmissing-noreturn** warns about functions that are candidates for 'noreturn' attribute. **-Wno-missing-noreturn** tells the compiler not to warn about functions that are candidates for 'noreturn' attribute.

-W[no-]missing-prototypes

(For C/C++ only) **-Wmissing-prototypes** warns about global funcs without prototypes. **-Wno-missing-prototypes** tells the compiler not to warn about global funcs without prototypes.

-W[no-]multichar

(For C/C++ only) **-Wmultichar** warns if a multi-character constant is used. **-Wno-multichar** tells the compiler not to warn if a multi-character constant is used.

-W[no-]nested-externs

(For C/C++ only) **-Wnested-externs** warns about externs not at file scope level. **-Wno-nested-externs** tells the compiler not to warn about externs not at file scope level.

-Wno-non-template-friend

(For C++ only) Do not warn about friend functions declared in templates.

-W[no-]non-virtual-dtor

(For C++ only) **-Wnon-virtual-dtor** will warn when a class declares a dtor (destructor) that should be virtual. **-Wno-non-virtual-dtor** tells the compiler not to warn when a class declares a dtor that should be

virtual.

-W[no-]old-style-cast

(For C/C++ only) **-Wold-style-cast** will warn when a C-style cast to a non-void type is used. **-Wno-old-style-cast** tells the compiler not to warn when a C-style cast to a non-void type is used.

-WOPT:

Specifies options that affect the global optimizer are enabled at **-O2** or above.

-WOPT:aggstr=N

This controls the aggressiveness of the strength reduction optimization performed by the scalar optimizer, in which induction expressions within a loop are replaced by temporaries that are incremented together with the loop variable. When strength reduction is overdone, the additional temporaries increase register pressure, resulting in excessive register spills that decrease performance. The value specified must be a positive integer value, which specifies the maximum number of induction expressions that will be strength-reduced across an index variable increment. When set at 0, strength reduction is only performed for non-trivial induction expressions. The default is 11.

-WOPT:const_pre=(ON|OFF)

When **OFF**, disables the placement optimization for loading constants to registers. Default is **ON**.

-WOPT:if_conv=(0|1|2)

Controls the optimization that translates simple IF statements to conditional move instructions in the target CPU. Setting to **0** suppresses this optimization. The value of **1** designates conservative if-conversion, in which the context around the IF statement is used in deciding whether to if-convert. The value of **2** enables aggressive if-conversion by causing it to be performed regardless of the context. The default is **1**.

-WOPT:ivar_pre=(ON|OFF)

When **OFF**, disables the partial redundancy elimination of indirect loads in the program. Default is **ON**.

-WOPT:mem_opnds=(ON|OFF)

Makes the scalar optimizer preserve any memory operands of arithmetic operations so as to help bring about subsumption of memory loads into the operands of arithmetic operations. Load subsumption is the combining of an arithmetic instruction and a memory load into one instruction. Default is **OFF**.

-WOPT:retype_expr=(ON|OFF)

Enables the optimization in the compiler that converts 64-bit address computation to use 32-bit arithmetic as much as possible. Default is **OFF**.

-WOPT:unroll=(0|1|2)

Control the unrolling of innermost loops in the scalar optimizer. Setting to 0 suppresses this unroller. The default is 1, which makes the scalar optimizer unroll only loops that contain IF statements. Setting to 2 makes the unrolling to also apply to loop bodies that are straight line code, which duplicates the unrolling done in the code generator, and is thus unnecessary. The default setting of 1 makes this unrolling complementary to what is done in the code generator. This unrolling is not affected by the unrolling options under the **-OPT** group.

-WOPT:val=(0|1|2)

Control the number of times the value-numbering optimization is performed in the global optimizer, with the default being **1**. This optimization tries to recognize expressions that will compute identical runtime values and changes the program to avoid re-computing them.

-W[no-]overloaded-virtual

(For C++ only) The **-Woverloaded-virtual** option will warn when a function declaration hides virtual functions. **-Wno-overloaded-virtual** tells the compiler not to warn when a function declaration hides virtual functions.

-W[no-]packed

(For C/C++ only) **-Wpacked** warns when packed attribute of a struct has no effect. **-Wno-packed** tells the compiler not to warn when packed attribute of a struct has no effect.

- W[no-]padded**
(For C/C++ only) **-Wpadded** warns when padding is included in a struct. **-Wno-padded** tells the compiler not to warn when padding is included in a struct.
- W[no-]parentheses**
(For C/C++ only) **-Wparentheses** warns about possible missing parentheses. **-Wno-parentheses** tells the compiler not to warn about possible missing parentheses.
- Wno-pmf-conversions**
(For C++ only) Do not warn about converting PMFs to plain pointers.
- W[no-]pointer-arith**
(For C/C++ only) **-Wpointer-arith** warns about function pointer arithmetic. **-Wno-pointer-arith** tells the compiler not to warn about function pointer arithmetic.
- W[no-]redundant-decls**
(For C/C++ only) **-Wredundant-decls** warns about multiple declarations of the same object. **-Wno-redundant-decls** tells the compiler not to warn about multiple declarations of the same object.
- W[no-]reorder**
(For C/C++ only) The **-Wreorder** option warns when reordering member initializers. **-Wno-reorder** tells the compiler not to warn when reordering member initializers.
- W[no-]return-type**
(For C/C++ only) **-Wreturn-type** warns when a function return type defaults to int. **-Wno-return-type** tells the compiler not to warn when a function return type defaults to int.
- W[no-]sequence-point**
(For C/C++ only) **-Wsequence-point** warns about code violating sequence point rules. **-Wno-sequence-point** tells the compiler not to warn about code violating sequence point rules.
- W[no-]shadow**
(For C/C++ only) **-Wshadow** warns when one local variable shadows another. **-Wno-shadow** tells the compiler not to warn when one local variable shadows another.
- W[no-]sign-compare**
(For C/C++ only) **-Wsign-compare** warns about signed/unsigned comparisons. **-Wno-sign-compare** tells the compiler not to warn about signed/unsigned comparisons.
- W[no-]sign-promo**
(For C/C++ only) The **-Wsign-promo** option warns when overload resolution promotes from unsigned to signed. **-Wno-sign-promo** tells the compiler not to warn when overload resolution promotes from unsigned to signed.
- W[no-]strict-aliasing**
(For C/C++ only) **-Wstrict-aliasing** warns about code that breaks strict aliasing rules. **-Wno-strict-aliasing** tells the compiler not to warn about code that breaks strict aliasing rules.
- W[no-]strict-prototypes**
(For C/C++ only) **-Wstrict-prototypes** warns about non-prototyped function decls. **-Wno-strict-prototypes** tells the compiler not to warn about non-prototyped function decls.
- W[no-]switch**
(For C/C++ only) **-Wswitch** warns when a switch statement is incorrectly indexed with an enum. **-Wno-switch** tells the compiler not to warn when a switch statement is incorrectly indexed with an enum.
- W[no-]system-headers**
(For C/C++ only) **-Wsystem-headers** prints warnings for constructs in system header files. **-Wno-system-headers** tells the compiler not to print warnings for constructs in system header files.
- W[no-]synth**
(For C++ only) The **-Wsynth** option warns about synthesis that is not backward compatible with cfront. **-Wno-synth** tells the compiler not to warn about synthesis that is not backwards compatible with cfront.

- W[no-]traditional**
(For C/C++ only) **-Wtraditional** warns about constructs whose meanings change in ANSI C. **-Wno-traditional** tells the compiler not to warn about constructs whose meanings change in ANSI C.
- W[no-]trigraphs**
(For C/C++ only) **-Wtrigraphs** warns when trigraphs are encountered. **-Wno-trigraphs** tells the compiler not to warn when trigraphs are encountered.
- W[no-]undef**
-Wundef warns if an undefined identifier appears in a #if directive. **-Wno-undef** tells the compiler not to warn if an undefined identifier appears in a #if directive.
- W[no-]uninitialized**
-Wuninitialized warns about uninitialized automatic variables. Because the analysis to find uninitialized variables is performed in the global optimizer invoked at -O2 or above, this option has no effect at **-O0** and **-O1**. **-Wno-uninitialized** tells the compiler not to warn about uninitialized automatic variables.
- W[no-]unknown-pragmas**
-Wunknown-pragmas warns when an unknown #pragma directive is encountered. **-Wno-unknown-pragmas** tells the compiler not to warn when an unknown #pragma directive is encountered.
- W[no-]unreachable-code**
-Wunreachable-code warns about code that will never be executed. **-Wno-unreachable-code** tells the compiler not to warn about code that will never be executed.
- W[no-]unused**
-Wunused warns when a variable is unused. **-Wno-unused** tells the compiler not to warn when a variable is unused.
- W[no-]unused-function**
-Wunused-function warns about unused static and inline functions. **-Wno-unused-function** tells the compiler not to warn about unused static and inline functions.
- W[no-]unused-label**
-Wunused-label warns about unused labels. **-Wno-unused-label** tells the compiler not to warn about unused labels.
- W[no-]unused-parameter**
-Wunused-parameter warns about unused function parameters. **-Wno-unused-parameter** tells the compiler not to warn about unused function parameters.
- W[no-]unused-value**
-Wunused-value warns about statements whose results are not used. **-Wno-unused-value** tells the compiler not to warn about statements whose results are not used.
- W[no-]unused-variable**
-Wunused-variable warns about local and static variables that are not used. **-Wno-unused-variable** tells the compiler not to warn about local and static variables that are not used.
- W[no-]write-strings**
-Wwrite-strings marks strings as 'const char*'. **-Wno-write-strings** tells the compiler not to mark strings as 'const char *'.
- Wnonnull**
(For C/C++ only) Warn when passing null to functions requiring non-null pointers.
- Wswitch-default**
(For C/C++ only) Warn when a switch statement has no default.
- Wswitch-enum**
(For C/C++ only) Warn when a switch statement is missing a case for an enum member.

- w** Suppress warning messages.
 - woff** Turn off named warnings
 - woffall**
Turn off all warnings.
 - woffoptions**
Turn off warnings about options.
 - woffnum**
Specify message numbers to suppress. Examples:
 - Specifying **-woff2026** suppresses message number 2026.
 - Specifying **-woff2026-2352** suppresses messages 2026 through 2352.
 - Specifying **-woff2026-2352,2400-2500** suppresses messages 2026 through 2352 and messages 2400 through 2500.
- In the message-level indicator, the message numbers appear after the dash.
- Yc,path**
Set the *path* in which to find the associated phase, using the same phase names as given in the **-W** option. The following characters can also be specified:
 - I** Specifies where to search for include files
 - S** Specifies where to search for startup files (**crt*.o**)
 - L** Specifies where to search for libraries
 - zerouf** Set uninitialized variables to zero. Affects local scalar and array variables and memory returned by `alloca()`. Does not affect the behavior of globals, `malloc()`ed memory, or Fortran common data.

ENVIRONMENT VARIABLES

F90_BOUNDS_CHECK_ABORT

(Fortran) Set to **YES**, causes the program to abort on the first bounds check violation.

F90_DUMP_MAP

(Fortran) When set to **YES**, if a segmentation fault occurs, print the current process's memory map before aborting. The memory map describes how the process's address space is allocated. The Fortran runtime will print the address of the segmentation fault; you can examine the memory map to see which mapped area was nearest to the fault address. This can help distinguish between program bugs that involve running out of stack space and null pointer dereferences. The memory map is displayed using the same format as the file `/proc/self/maps`.

FILENV

The location of the **assign** file. See the **assign** (1) man page for more details.

FTN_SUPPRESS_REPEATS

(Fortran) Output multiple values instead of using the repeat factor, used at runtime.

NLSPATH

(Fortran) Flags for runtime and compile-time messages.

PSC_CFLAGS

(C) Flags to pass to the C compiler, `pathcc`.

PSC_COMPILER_DEFAULTS_PATH

Specifies a path or colon-separated list of paths, designating where the compiler is to look for the **compiler.defaults(5)** file. If the environment variable is set, the path `/opt/pathscale/etc` will not be used. If the file cannot be found, then no defaults file will be used, even if one is present in `/opt/pathscale/etc`.

PSC_PROBLEM_REPORT_DIR

Name a directory in which to save problem reports and preprocessed source files, if the compiler encounters an internal error. If not specified, the directory used is **\$HOME/.ekopath-bugs**.

PSC_CXXFLAGS

(C++) Flags to pass to the C++ compiler, pathCC.

PSC_FFLAGS

(Fortran) Flags to pass to the Fortran compiler, pathf95.

PSC_GENFLAGS

Generic flags passed to all compilers.

PSC_STACK_LIMIT

(Fortran) Controls the stack size limit the Fortran runtime attempts to use. This string takes the format of a floating-point number, optionally followed by one of the characters "k" (for units of 1024 bytes), "m" (for units of 1048576 bytes), "g" (for units of 1073741824 bytes), or "%" (to specify a percentage of physical memory). If the specifier is following by the string "/cpu", the limit is divided by the number of CPUs the system has. For example, a limit of "1.5g" specifies that the Fortran runtime will use no more than 1.5 gigabytes (GB) of stack. On a system with 2GB of physical memory, a limit of "90%/cpu" will use no more than 0.9GB of stack ($2/2 * 0.90$).

PSC_STACK_VERBOSE

(Fortran) If this environment variable is set, the Fortran runtime will print detailed information about how it is computing the stack size limit to use.

Standard OpenMP Runtime Environment Variables

These environment variables can be used with OpenMP in either Fortran or C and C++.

OMP_DYNAMIC

Enables or disables dynamic adjustment of the number of threads available for execution. Default is **FALSE**, since this mechanism is not supported.

OMP_NESTED

Enables or disables nested parallelism. Default is **FALSE**.

OMP_SCHEDULE

This environment variable only applies to DO and PARALLEL_DO directives that have schedule type RUNTIME. Type can be STATIC, DYNAMIC, or GUIDED. Default is **STATIC**, with no chunk size specified.

OMP_NUM_THREADS

Set the number of threads to use during execution. Default is number of CPUs in the machine.

PathScale OpenMP Environment Variables

These environment variables can be used with OpenMP in both Fortran and C and C++, except as indicated.

PSC_OMP_AFFINITY

When **TRUE**, the operating system's affinity mechanism (where available) is used to assign threads to CPUs, otherwise no affinity assignments are made. The default value is **TRUE**.

PSC_OMP_AFFINITY_GLOBAL

This environment variable controls where thread global ID or local ID values are used when assigning threads to CPUs. The default is **TRUE** so that global ID values are used for calculating thread assignments.

PSC_OMP_AFFINITY_MAP

This environment variable allows the mapping from threads to CPUs to be fully specified by the user. It must be set to a list of CPU identifiers separated by commas. The list must contain at least one CPU identifier, and entries in the list beyond the maximum number of threads supported by the implementation (256) are ignored. Each CPU identifier is a decimal number between 0 and one

less than the number of CPUs in the system (inclusive).

The implementation generates a mapping table that enumerates the mapping from each thread to CPUs. The CPU identifiers in the **PSC_OMP_AFFINITY_MAP** list are inserted in the mapping table starting at the index for thread 0 and increasing upwards. If the list is shorter than the maximum number of threads, then it is simply repeated over and over again until there is a mapping for each thread. This repeat feature allows short lists to be used to specify repetitive thread mappings for all threads.

PSC_OMP_CPU_STRIDE

This specifies the striding factor used when mapping threads to CPUs. It takes an integer value in the range of 0 to the number of CPUs (inclusive). The default is a stride of **1**, which causes the threads to be linearly mapped to consecutive CPUs. When there are more threads than CPUs the mapping wraps around giving a round-robin allocation of threads to CPUs. The behavior for a stride of 0 is the same as a stride of 1.

PSC_OMP_CPU_OFFSET

This specifies an integer value that is used to offset the CPU assignments for the set of threads. It takes an integer value in the range of 0 to the number of CPUs (inclusive). When a thread is mapped to a CPU, this offset is added onto the CPU number calculated after **PSC_OMP_CPU_STRIDE** has been applied. If the resulting value is greater than the number of CPUs, then the remainder is used from the division of this value by the number of CPUs.

PSC_OMP_GUARD_SIZE

This environment variable specifies the size in bytes of a guard area that is placed below pthread stacks. This guard area is in addition to any guard pages created by your O/S.

PSC_OMP_GUIDED_CHUNK_DIVISOR

The value of **PSC_OMP_GUIDED_CHUNK_DIVISOR** is used to divide down the chunk size assigned by the guided scheduling algorithm.

PSC_OMP_GUIDED_CHUNK_MAX

This is the maximum chunk size that will be used by the loop scheduler for guided scheduling.

PSC_OMP_LOCK_SPIN

This chooses the locking mechanism used by critical sections and OMP locks.

PSC_OMP_SILENT

If you set **PSC_OMP_SILENT** to anything, then warning and debug messages from the libopenmp library are inhibited.

PSC_OMP_STACK_SIZE

(Fortran) Stack size specification follows the syntax in described in the *OpenMP in Fortran* section of *QLogic PathScale Compiler Suite User Guide*.

PSC_OMP_STATIC_FAIR

This determines the default static scheduling policy when no chunk size is specified. It is discussed in the *OpenMP in Fortran* section of *QLogic PathScale Compiler Suite User Guide*.

PSC_OMP_THREAD_SPIN

This takes a numeric value and sets the number of times that the spin loops will spin at user-level before falling back to O/S schedule/reschedule mechanisms.

COPYRIGHT

Copyright (C) 2006, 2007 QLogic Corp. All Rights Reserved.

Copyright (C) 2003, 2004, 2005, 2006 PathScale, Inc. All Rights Reserved.

SEE ALSO

pathcc(1), **pathCC(1)**, **pathf95(1)**, **compiler.defaults(5)**, **pathopt2(1)**, **assign(1)**, **explain(1)**, **fsymlist(1)**, **pathscale_intro(7)**, **pathdb(1)**

QLogic PathScale Compiler Suite and Subscription Manager Install Guide

QLogic PathScale Compiler Suite User Guide

QLogic PathScale Compiler Suite Support Guide

QLogic PathScale Debugger User Guide

Online documentation available at <http://www.pathscale.com/docs.html>