# gfotran编译参数

## Fotran Language Options

See [Options controlling Fortran dialect](https://gcc.gnu.org/onlinedocs/gfortran/Fortran-Dialect-Options.html#Fortran-Dialect-Options).

-fall-intrinsics -fallow-argument-mismatch -fallow-invalid-boz

-fbackslash -fcray-pointer -fd-lines-as-code -fd-lines-as-comments

-fdec -fdec-char-conversions -fdec-structure -fdec-intrinsic-ints

-fdec-static -fdec-math -fdec-include -fdec-format-defaults

-fdec-blank-format-item -fdefault-double-8 -fdefault-integer-8

-fdefault-real-8 -fdefault-real-10 -fdefault-real-16 -fdollar-ok

-ffixed-line-length-*n* -ffixed-line-length-none -fpad-source

-ffree-form -ffree-line-length-*n* -ffree-line-length-none

-fimplicit-none -finteger-4-integer-8 -fmax-identifier-length

-fmodule-private -ffixed-form -fno-range-check -fopenacc -fopenmp

-freal-4-real-10 -freal-4-real-16 -freal-4-real-8 -freal-8-real-10

-freal-8-real-16 -freal-8-real-4 -std=*std* -ftest-forall-temp

 Options controlling Fortran dialect

The following options control the details of the Fortran dialect accepted by the compiler:

-ffree-form

-ffixed-form

Specify the layout used by the source file. The free form layout was introduced in Fortran 90. Fixed form was traditionally used in older Fortran programs. When neither option is specified, the source form is determined by the file extension.

-fall-intrinsics

This option causes all intrinsic procedures (including the GNU-specific extensions) to be accepted. This can be useful with -std=f95 to force standard-compliance but get access to the full range of intrinsics available with gfortran. As a consequence, -Wintrinsics-std will be ignored and no user-defined procedure with the same name as any intrinsic will be called except when it is explicitly declared EXTERNAL.

-fallow-argument-mismatch

Some code contains calls to external procedures with mismatches between the calls and the procedure definition, or with mismatches between different calls. Such code is non-conforming, and will usually be flagged with an error. This options degrades the error to a warning, which can only be disabled by disabling all warnings via -w. Only a single occurrence per argument is flagged by this warning. -fallow-argument-mismatch is implied by -std=legacy.

Using this option is *strongly* discouraged. It is possible to provide standard-conforming code which allows different types of arguments by using an explicit interface and TYPE(\*).

-fallow-invalid-boz

A BOZ literal constant can occur in a limited number of contexts in standard conforming Fortran. This option degrades an error condition to a warning, and allows a BOZ literal constant to appear where the Fortran standard would otherwise prohibit its use.

-fd-lines-as-code

-fd-lines-as-comments

Enable special treatment for lines beginning with d or D in fixed form sources. If the -fd-lines-as-code option is given they are treated as if the first column contained a blank. If the -fd-lines-as-comments option is given, they are treated as comment lines.

-fdec

DEC compatibility mode. Enables extensions and other features that mimic the default behavior of older compilers (such as DEC). These features are non-standard and should be avoided at all costs. For details on GNU Fortran’s implementation of these extensions see the full documentation.

Other flags enabled by this switch are: -fdollar-ok -fcray-pointer -fdec-char-conversions -fdec-structure -fdec-intrinsic-ints -fdec-static -fdec-math -fdec-include -fdec-blank-format-item -fdec-format-defaults

If -fd-lines-as-code/-fd-lines-as-comments are unset, then -fdec also sets -fd-lines-as-comments.

-fdec-char-conversions

Enable the use of character literals in assignments and DATA statements for non-character variables.

-fdec-structure

Enable DEC STRUCTURE and RECORD as well as UNION, MAP, and dot (’.’) as a member separator (in addition to ’%’). This is provided for compatibility only; Fortran 90 derived types should be used instead where possible.

-fdec-intrinsic-ints

Enable B/I/J/K kind variants of existing integer functions (e.g. BIAND, IIAND, JIAND, etc...). For a complete list of intrinsics see the full documentation.

-fdec-math

Enable legacy math intrinsics such as COTAN and degree-valued trigonometric functions (e.g. TAND, ATAND, etc...) for compatability with older code.

-fdec-static

Enable DEC-style STATIC and AUTOMATIC attributes to explicitly specify the storage of variables and other objects.

-fdec-include

Enable parsing of INCLUDE as a statement in addition to parsing it as INCLUDE line. When parsed as INCLUDE statement, INCLUDE does not have to be on a single line and can use line continuations.

-fdec-format-defaults

Enable format specifiers F, G and I to be used without width specifiers, default widths will be used instead.

-fdec-blank-format-item

Enable a blank format item at the end of a format specification i.e. nothing following the final comma.

-fdollar-ok

Allow ‘$’ as a valid non-first character in a symbol name. Symbols that start with ‘$’ are rejected since it is unclear which rules to apply to implicit typing as different vendors implement different rules. Using ‘$’ in IMPLICIT statements is also rejected.

-fbackslash

Change the interpretation of backslashes in string literals from a single backslash character to “C-style” escape characters. The following combinations are expanded \a, \b, \f, \n, \r, \t, \v, \\, and \0 to the ASCII characters alert, backspace, form feed, newline, carriage return, horizontal tab, vertical tab, backslash, and NUL, respectively. Additionally, \x*nn*, \u*nnnn* and \U*nnnnnnnn* (where each *n* is a hexadecimal digit) are translated into the Unicode characters corresponding to the specified code points. All other combinations of a character preceded by \ are unexpanded.

-fmodule-private

Set the default accessibility of module entities to PRIVATE. Use-associated entities will not be accessible unless they are explicitly declared as PUBLIC.

-ffixed-line-length-*n*

Set column after which characters are ignored in typical fixed-form lines in the source file, and, unless -fno-pad-source, through which spaces are assumed (as if padded to that length) after the ends of short fixed-form lines.

Popular values for *n* include 72 (the standard and the default), 80 (card image), and 132 (corresponding to “extended-source” options in some popular compilers). *n* may also be ‘none’, meaning that the entire line is meaningful and that continued character constants never have implicit spaces appended to them to fill out the line. -ffixed-line-length-0 means the same thing as -ffixed-line-length-none.

-fno-pad-source

By default fixed-form lines have spaces assumed (as if padded to that length) after the ends of short fixed-form lines. This is not done either if -ffixed-line-length-0, -ffixed-line-length-none or if -fno-pad-source option is used. With any of those options continued character constants never have implicit spaces appended to them to fill out the line.

-ffree-line-length-*n*

Set column after which characters are ignored in typical free-form lines in the source file. The default value is 132. *n* may be ‘none’, meaning that the entire line is meaningful. -ffree-line-length-0 means the same thing as -ffree-line-length-none.

-fmax-identifier-length=*n*

Specify the maximum allowed identifier length. Typical values are 31 (Fortran 95) and 63 (Fortran 2003 and Fortran 2008).

-fimplicit-none

Specify that no implicit typing is allowed, unless overridden by explicit IMPLICIT statements. This is the equivalent of adding implicit none to the start of every procedure.

-fcray-pointer

Enable the Cray pointer extension, which provides C-like pointer functionality.

-fopenacc

Enable the OpenACC extensions. This includes OpenACC !$acc directives in free form and c$acc, \*$acc and !$acc directives in fixed form, !$ conditional compilation sentinels in free form and c$, \*$ and !$ sentinels in fixed form, and when linking arranges for the OpenACC runtime library to be linked in.

-fopenmp

Enable the OpenMP extensions. This includes OpenMP !$omp directives in free form and c$omp, \*$omp and !$omp directives in fixed form, !$ conditional compilation sentinels in free form and c$, \*$ and !$ sentinels in fixed form, and when linking arranges for the OpenMP runtime library to be linked in. The option -fopenmp implies -frecursive.

-fno-range-check

Disable range checking on results of simplification of constant expressions during compilation. For example, GNU Fortran will give an error at compile time when simplifying a = 1. / 0. With this option, no error will be given and a will be assigned the value +Infinity. If an expression evaluates to a value outside of the relevant range of [-HUGE():HUGE()], then the expression will be replaced by -Inf or +Inf as appropriate. Similarly, DATA i/Z'FFFFFFFF'/ will result in an integer overflow on most systems, but with -fno-range-check the value will “wrap around” and i will be initialized to *-1* instead.

-fdefault-integer-8

Set the default integer and logical types to an 8 byte wide type. This option also affects the kind of integer constants like 42. Unlike -finteger-4-integer-8, it does not promote variables with explicit kind declaration.

-fdefault-real-8

Set the default real type to an 8 byte wide type. This option also affects the kind of non-double real constants like 1.0. This option promotes the default width of DOUBLE PRECISION and double real constants like 1.d0 to 16 bytes if possible. If -fdefault-double-8 is given along with fdefault-real-8, DOUBLE PRECISION and double real constants are not promoted. Unlike -freal-4-real-8, fdefault-real-8 does not promote variables with explicit kind declarations.

-fdefault-real-10

Set the default real type to an 10 byte wide type. This option also affects the kind of non-double real constants like 1.0. This option promotes the default width of DOUBLE PRECISION and double real constants like 1.d0 to 16 bytes if possible. If -fdefault-double-8 is given along with fdefault-real-10, DOUBLE PRECISION and double real constants are not promoted. Unlike -freal-4-real-10, fdefault-real-10 does not promote variables with explicit kind declarations.

-fdefault-real-16

Set the default real type to an 16 byte wide type. This option also affects the kind of non-double real constants like 1.0. This option promotes the default width of DOUBLE PRECISION and double real constants like 1.d0 to 16 bytes if possible. If -fdefault-double-8 is given along with fdefault-real-16, DOUBLE PRECISION and double real constants are not promoted. Unlike -freal-4-real-16, fdefault-real-16 does not promote variables with explicit kind declarations.

-fdefault-double-8

Set the DOUBLE PRECISION type and double real constants like 1.d0 to an 8 byte wide type. Do nothing if this is already the default. This option prevents -fdefault-real-8, -fdefault-real-10, and -fdefault-real-16, from promoting DOUBLE PRECISION and double real constants like 1.d0 to 16 bytes.

-finteger-4-integer-8

Promote all INTEGER(KIND=4) entities to an INTEGER(KIND=8) entities. If KIND=8 is unavailable, then an error will be issued. This option should be used with care and may not be suitable for your codes. Areas of possible concern include calls to external procedures, alignment in EQUIVALENCE and/or COMMON, generic interfaces, BOZ literal constant conversion, and I/O. Inspection of the intermediate representation of the translated Fortran code, produced by -fdump-tree-original, is suggested.

-freal-4-real-8

-freal-4-real-10

-freal-4-real-16

-freal-8-real-4

-freal-8-real-10

-freal-8-real-16

Promote all REAL(KIND=M) entities to REAL(KIND=N) entities. If REAL(KIND=N) is unavailable, then an error will be issued. The -freal-4- flags also affect the default real kind and the -freal-8- flags also the double-precision real kind. All other real-kind types are unaffected by this option. The promotion is also applied to real literal constants of default and double-precision kind and a specified kind number of 4 or 8, respectively. However, -fdefault-real-8, -fdefault-real-10, -fdefault-real-10, and -fdefault-double-8 take precedence for the default and double-precision real kinds, both for real literal constants and for declarations without a kind number. Note that for REAL(KIND=KIND(1.0)) the literal may get promoted and then the result may get promoted again. These options should be used with care and may not be suitable for your codes. Areas of possible concern include calls to external procedures, alignment in EQUIVALENCE and/or COMMON, generic interfaces, BOZ literal constant conversion, and I/O and calls to intrinsic procedures when passing a value to the kind= dummy argument. Inspection of the intermediate representation of the translated Fortran code, produced by -fdump-fortran-original or -fdump-tree-original, is suggested.

-std=*std*

Specify the standard to which the program is expected to conform, which may be one of ‘f95’, ‘f2003’, ‘f2008’, ‘f2018’, ‘gnu’, or ‘legacy’. The default value for *std* is ‘gnu’, which specifies a superset of the latest Fortran standard that includes all of the extensions supported by GNU Fortran, although warnings will be given for obsolete extensions not recommended for use in new code. The ‘legacy’ value is equivalent but without the warnings for obsolete extensions, and may be useful for old non-standard programs. The ‘f95’, ‘f2003’, ‘f2008’, and ‘f2018’ values specify strict conformance to the Fortran 95, Fortran 2003, Fortran 2008 and Fortran 2018 standards, respectively; errors are given for all extensions beyond the relevant language standard, and warnings are given for the Fortran 77 features that are permitted but obsolescent in later standards. The deprecated option ‘-std=f2008ts’ acts as an alias for ‘-std=f2018’. It is only present for backwards compatibility with earlier gfortran versions and should not be used any more.

-ftest-forall-temp

Enhance test coverage by forcing most forall assignments to use temporary.

## Preprocessing Options

See [Enable and customize preprocessing](https://gcc.gnu.org/onlinedocs/gfortran/Preprocessing-Options.html#Preprocessing-Options).

-A-question[=answer]

-Aquestion=answer -C -CC -Dmacro[=defn]

-H -P

-Umacro -cpp -dD -dI -dM -dN -dU -fworking-directory

-imultilib dir

-iprefix file -iquote -isysroot dir -isystem dir -nocpp

-nostdinc

-undef

Enable and customize preprocessing

Preprocessor related options. See section [Preprocessing and conditional compilation](https://gcc.gnu.org/onlinedocs/gfortran/Preprocessing-and-conditional-compilation.html#Preprocessing-and-conditional-compilation) for more detailed information on preprocessing in gfortran.

-cpp

-nocpp

Enable preprocessing. The preprocessor is automatically invoked if the file extension is .fpp, .FPP, .F, .FOR, .FTN, .F90, .F95, .F03 or .F08. Use this option to manually enable preprocessing of any kind of Fortran file.

To disable preprocessing of files with any of the above listed extensions, use the negative form: -nocpp.

The preprocessor is run in traditional mode. Any restrictions of the file-format, especially the limits on line length, apply for preprocessed output as well, so it might be advisable to use the -ffree-line-length-none or -ffixed-line-length-none options.

-dM

Instead of the normal output, generate a list of '#define' directives for all the macros defined during the execution of the preprocessor, including predefined macros. This gives you a way of finding out what is predefined in your version of the preprocessor. Assuming you have no file foo.f90, the command

touch foo.f90; gfortran -cpp -E -dM foo.f90

will show all the predefined macros.

-dD

Like -dM except in two respects: it does not include the predefined macros, and it outputs both the #define directives and the result of preprocessing. Both kinds of output go to the standard output file.

-dN

Like -dD, but emit only the macro names, not their expansions.

-dU

Like dD except that only macros that are expanded, or whose definedness is tested in preprocessor directives, are output; the output is delayed until the use or test of the macro; and '#undef' directives are also output for macros tested but undefined at the time.

-dI

Output '#include' directives in addition to the result of preprocessing.

-fworking-directory

Enable generation of linemarkers in the preprocessor output that will let the compiler know the current working directory at the time of preprocessing. When this option is enabled, the preprocessor will emit, after the initial linemarker, a second linemarker with the current working directory followed by two slashes. GCC will use this directory, when it is present in the preprocessed input, as the directory emitted as the current working directory in some debugging information formats. This option is implicitly enabled if debugging information is enabled, but this can be inhibited with the negated form -fno-working-directory. If the -P flag is present in the command line, this option has no effect, since no #line directives are emitted whatsoever.

-idirafter *dir*

Search *dir* for include files, but do it after all directories specified with -I and the standard system directories have been exhausted. *dir* is treated as a system include directory. If dir begins with =, then the = will be replaced by the sysroot prefix; see --sysroot and -isysroot.

-imultilib *dir*

Use *dir* as a subdirectory of the directory containing target-specific C++ headers.

-iprefix *prefix*

Specify *prefix* as the prefix for subsequent -iwithprefix options. If the *prefix* represents a directory, you should include the final '/'.

-isysroot *dir*

This option is like the --sysroot option, but applies only to header files. See the --sysroot option for more information.

-iquote *dir*

Search *dir* only for header files requested with #include "file"; they are not searched for #include <file>, before all directories specified by -I and before the standard system directories. If *dir* begins with =, then the = will be replaced by the sysroot prefix; see --sysroot and -isysroot.

-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. If *dir* begins with =, then the = will be replaced by the sysroot prefix; see --sysroot and -isysroot.

-nostdinc

Do not search the standard system directories for header files. Only the directories you have specified with -I options (and the directory of the current file, if appropriate) are searched.

-undef

Do not predefine any system-specific or GCC-specific macros. The standard predefined macros remain defined.

-A*predicate*=*answer*

Make an assertion with the predicate *predicate* and answer *answer*. This form is preferred to the older form -A predicate(answer), which is still supported, because it does not use shell special characters.

-A-*predicate*=*answer*

Cancel an assertion with the predicate *predicate* and answer *answer*.

-C

Do not discard comments. All comments are passed through to the output file, except for comments in processed directives, which are deleted along with the directive.

You should be prepared for side effects when using -C; it causes the preprocessor to treat comments as tokens in their own right. For example, comments appearing at the start of what would be a directive line have the effect of turning that line into an ordinary source line, since the first token on the line is no longer a '#'.

Warning: this currently handles C-Style comments only. The preprocessor does not yet recognize Fortran-style comments.

-CC

Do not discard comments, including during macro expansion. This is like -C, except that comments contained within macros are also passed through to the output file where the macro is expanded.

In addition to the side-effects of the -C option, the -CC option causes all C++-style comments inside a macro to be converted to C-style comments. This is to prevent later use of that macro from inadvertently commenting out the remainder of the source line. The -CC option is generally used to support lint comments.

Warning: this currently handles C- and C++-Style comments only. The preprocessor does not yet recognize Fortran-style comments.

-D*name*

Predefine name as a macro, with definition 1.

-D*name*=*definition*

The contents of *definition* are tokenized and processed as if they appeared during translation phase three in a '#define' directive. In particular, the definition will be truncated by embedded newline characters.

If you are invoking the preprocessor from a shell or shell-like program you may need to use the shell’s quoting syntax to protect characters such as spaces that have a meaning in the shell syntax.

If you wish to define a function-like macro on the command line, write its argument list with surrounding parentheses before the equals sign (if any). Parentheses are meaningful to most shells, so you will need to quote the option. With sh and csh, -D'name(args...)=definition' works.

-D and -U options are processed in the order they are given on the command line. All -imacros file and -include file options are processed after all -D and -U options.

-H

Print the name of each header file used, in addition to other normal activities. Each name is indented to show how deep in the '#include' stack it is.

-P

Inhibit generation of linemarkers in the output from the preprocessor. This might be useful when running the preprocessor on something that is not C code, and will be sent to a program which might be confused by the linemarkers.

-U*name*

Cancel any previous definition of *name*, either built in or provided with a -D option.

## Error and Warning Options

See [Options to request or suppress errors and warnings](https://gcc.gnu.org/onlinedocs/gfortran/Error-and-Warning-Options.html#Error-and-Warning-Options).

-Waliasing -Wall -Wampersand -Warray-bounds

-Wc-binding-type -Wcharacter-truncation -Wconversion

-Wdo-subscript -Wfunction-elimination -Wimplicit-interface

-Wimplicit-procedure -Wintrinsic-shadow -Wuse-without-only

-Wintrinsics-std -Wline-truncation -Wno-align-commons

-Wno-overwrite-recursive -Wno-tabs -Wreal-q-constant -Wsurprising

-Wunderflow -Wunused-parameter -Wrealloc-lhs -Wrealloc-lhs-all

-Wfrontend-loop-interchange -Wtarget-lifetime -fmax-errors=*n*

-fsyntax-only -pedantic

-pedantic-errors

## Debugging Options

See [Options for debugging your program or GNU Fortran](https://gcc.gnu.org/onlinedocs/gfortran/Debugging-Options.html#Debugging-Options).

-fbacktrace -fdump-fortran-optimized -fdump-fortran-original

-fdebug-aux-vars -fdump-fortran-global -fdump-parse-tree -ffpe-trap=*list*

-ffpe-summary=*list*

Options for debugging your program or GNU Fortran

GNU Fortran has various special options that are used for debugging either your program or the GNU Fortran compiler.

-fdump-fortran-original

Output the internal parse tree after translating the source program into internal representation. This option is mostly useful for debugging the GNU Fortran compiler itself. The output generated by this option might change between releases. This option may also generate internal compiler errors for features which have only recently been added.

-fdump-fortran-optimized

Output the parse tree after front-end optimization. Mostly useful for debugging the GNU Fortran compiler itself. The output generated by this option might change between releases. This option may also generate internal compiler errors for features which have only recently been added.

-fdump-parse-tree

Output the internal parse tree after translating the source program into internal representation. Mostly useful for debugging the GNU Fortran compiler itself. The output generated by this option might change between releases. This option may also generate internal compiler errors for features which have only recently been added. This option is deprecated; use -fdump-fortran-original instead.

-fdebug-aux-vars

Renames internal variables created by the gfortran front end and makes them accessible to a debugger. The name of the internal variables then start with upper-case letters followed by an underscore. This option is useful for debugging the compiler’s code generation together with -fdump-tree-original and enabling debugging of the executable program by using -g or -ggdb3.

-fdump-fortran-global

Output a list of the global identifiers after translating into middle-end representation. Mostly useful for debugging the GNU Fortran compiler itself. The output generated by this option might change between releases. This option may also generate internal compiler errors for features which have only recently been added.

-ffpe-trap=*list*

Specify a list of floating point exception traps to enable. On most systems, if a floating point exception occurs and the trap for that exception is enabled, a SIGFPE signal will be sent and the program being aborted, producing a core file useful for debugging. *list* is a (possibly empty) comma-separated list of the following exceptions: ‘invalid’ (invalid floating point operation, such as SQRT(-1.0)), ‘zero’ (division by zero), ‘overflow’ (overflow in a floating point operation), ‘underflow’ (underflow in a floating point operation), ‘inexact’ (loss of precision during operation), and ‘denormal’ (operation performed on a denormal value). The first five exceptions correspond to the five IEEE 754 exceptions, whereas the last one (‘denormal’) is not part of the IEEE 754 standard but is available on some common architectures such as x86.

The first three exceptions (‘invalid’, ‘zero’, and ‘overflow’) often indicate serious errors, and unless the program has provisions for dealing with these exceptions, enabling traps for these three exceptions is probably a good idea.

If the option is used more than once in the command line, the lists will be joined: ’ffpe-trap=*list1* ffpe-trap=*list2*’ is equivalent to ffpe-trap=*list1*,*list2*.

Note that once enabled an exception cannot be disabled (no negative form).

Many, if not most, floating point operations incur loss of precision due to rounding, and hence the ffpe-trap=inexact is likely to be uninteresting in practice.

By default no exception traps are enabled.

-ffpe-summary=*list*

Specify a list of floating-point exceptions, whose flag status is printed to ERROR\_UNIT when invoking STOP and ERROR STOP. *list* can be either ‘none’, ‘all’ or a comma-separated list of the following exceptions: ‘invalid’, ‘zero’, ‘overflow’, ‘underflow’, ‘inexact’ and ‘denormal’. (See -ffpe-trap for a description of the exceptions.)

If the option is used more than once in the command line, only the last one will be used.

By default, a summary for all exceptions but ‘inexact’ is shown.

-fno-backtrace

When a serious runtime error is encountered or a deadly signal is emitted (segmentation fault, illegal instruction, bus error, floating-point exception, and the other POSIX signals that have the action ‘core’), the Fortran runtime library tries to output a backtrace of the error. -fno-backtrace disables the backtrace generation. This option only has influence for compilation of the Fortran main program.

## Directory Options

See [Options for directory search](https://gcc.gnu.org/onlinedocs/gfortran/Directory-Options.html#Directory-Options).

-I*dir* -J*dir* -fintrinsic-modules-path *dir*

## Link Options

See [Options for influencing the linking step](https://gcc.gnu.org/onlinedocs/gfortran/Link-Options.html#Link-Options).

-static-libgfortran

## Runtime Options

See [Options for influencing runtime behavior](https://gcc.gnu.org/onlinedocs/gfortran/Runtime-Options.html#Runtime-Options).

-fconvert=*conversion* -fmax-subrecord-length=*length*

-frecord-marker=*length* -fsign-zero

2.8 Influencing runtime behavior

These options affect the runtime behavior of programs compiled with GNU Fortran.

-fconvert=*conversion*

Specify the representation of data for unformatted files. Valid values for conversion are: ‘native’, the default; ‘swap’, swap between big- and little-endian; ‘big-endian’, use big-endian representation for unformatted files; ‘little-endian’, use little-endian representation for unformatted files.

This option has an effect only when used in the main program. The CONVERT specifier and the GFORTRAN\_CONVERT\_UNIT environment variable override the default specified by -fconvert.

-frecord-marker=*length*

Specify the length of record markers for unformatted files. Valid values for *length* are 4 and 8. Default is 4. *This is different from previous versions of gfortran*, which specified a default record marker length of 8 on most systems. If you want to read or write files compatible with earlier versions of gfortran, use -frecord-marker=8.

-fmax-subrecord-length=*length*

Specify the maximum length for a subrecord. The maximum permitted value for length is 2147483639, which is also the default. Only really useful for use by the gfortran testsuite.

-fsign-zero

When enabled, floating point numbers of value zero with the sign bit set are written as negative number in formatted output and treated as negative in the SIGN intrinsic. -fno-sign-zero does not print the negative sign of zero values (or values rounded to zero for I/O) and regards zero as positive number in the SIGN intrinsic for compatibility with Fortran 77. The default is -fsign-zero.

## Interoperability Options

See [Options for interoperability](https://gcc.gnu.org/onlinedocs/gfortran/Interoperability-Options.html#Interoperability-Options).

-fc-prototypes -fc-prototypes-external

## Code Generation Options

See [Options for code generation conventions](https://gcc.gnu.org/onlinedocs/gfortran/Code-Gen-Options.html#Code-Gen-Options).

-faggressive-function-elimination -fblas-matmul-limit=*n*

-fbounds-check -ftail-call-workaround -ftail-call-workaround=*n*

-fcheck-array-temporaries

-fcheck=*<all|array-temps|bits|bounds|do|mem|pointer|recursion>*

-fcoarray=*<none|single|lib>* -fexternal-blas -ff2c

-ffrontend-loop-interchange -ffrontend-optimize

-finit-character=*n* -finit-integer=*n* -finit-local-zero

-finit-derived -finit-logical=*<true|false>*

-finit-real=*<zero|inf|-inf|nan|snan>*

-finline-matmul-limit=*n*

-finline-arg-packing -fmax-array-constructor=*n*

-fmax-stack-var-size=*n* -fno-align-commons -fno-automatic

-fno-protect-parens -fno-underscoring -fsecond-underscore

-fpack-derived -frealloc-lhs -frecursive -frepack-arrays

-fshort-enums -fstack-arrays

# ifort 中的几个有用的调试程序选项

-WB       turn a compile-time bounds check into a warning

-Wcheck   enable more strict diagnostics

-Winline  enable inline diagnostics

-[no]traceback

          specify whether the compiler generates PC correlation data used to

          display a symbolic traceback rather than a hexadecimal traceback at

          runtime failure

-heap-arrays [n]

          temporary arrays of minimum size n (in kilobytes) are allocated in

          heap memory rather than on the stack.  If n is not specified,

          all temporary arrays are allocated in heap memory.

-fp-stack-check

          enable fp stack checking after every function/procedure call

-warn <keyword>

          specifies the level of warning messages issued

            keywords: all, none (same as -nowarn)

                      [no]alignments, [no]declarations, [no]errors,

                      [no]general, [no]ignore\_loc, [no]interfaces,

                      [no]stderrors, [no]truncated\_source, [no]uncalled,

                      [no]unused, [no]usage

————————————————

版权声明：本文为CSDN博主「cepheid」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/cepheid/article/details/5110272