There is little standardization in the choice of compiler options from one compiler to another, though a couple (`-o` and `-c`, for example) are the same across all platforms. Even the `-o` option differs slightly in meaning from one system to another, however. If there is *any* reliable documentation, it's what was supplied with your compiler. This doesn't help you, of course, if you have a *Makefile* from some unfamiliar machine and you're trying to figure out what the options there mean. This table should fill that gap: if you find an option, chances are this table will help you guess what it means. If you're looking for a way to tell *your* compiler what to do, read its documentation.

This appendix provides the following comparative references between the GNU, SGI IRIX, SCO UNIX, Solaris, SunOS, System V.3, System V.4 and XENIX versions of the C compiler control program and the C preprocessor.

- Flags used by the C compiler control program (*cc* or *gcc*), starting after this section.
- *gcc*-specific options specifying dialect, starting on page 405.
- *gcc*-specific debugging options, starting on page 406.
- *gcc*-specific warning options, starting on page 407. We discuss the more important warnings in , starting on page .
- Flags used by the C preprocessor *cpp*, starting on page 410.

C compiler options

`-a` *(gcc, SunOS)*

Generate extra code to write profile information for *tcov*.

`-a` *align* *(some MS-DOS compilers)*

Align in structs to *align* boundary.

`-A` *(SVR3)*

Linker output should be an absolute file (i.e. the opposite of the `-r` option).

`-A` *(gcc, SVR4)*

`-A`*question*`(`*answer*`)` asserts that the answer to *question* is *answer*. This can used with the preprocessor conditional *#if #question(answer)*.

`-A-` *(gcc, SVR4)*

Disable standard assertions. In addition, SVR4 *cc* undefines all standard macros except those beginning with __.

`-acpp` *(SGI)*

Use alternative *cpp* based on GNU *cpp*.

`-align` *block* *(SunOS)*

Force the global bss symbol *block* to be aligned to the beginning of a page.

`-ansi` *(gcc, SGI)*

Enforce strict ANSI compatibility.

`-ansiposix` *(SGI)*

Enforce strict ANSI compatibility and define `_POSIX_SOURCE`.

`-B` *(gcc)*

Specify the path of the directory from which the compiler control program *gcc* should start the individual passes of the compiler.

`-B dynamic` *(SunOS, SVR4)*

Dynamic linking: tell the linker to search for library files named *libfoo.so* and then *libfoo.a* when passed the option `-lfoo`.

`-B static` *(SunOS, SVR4)*

Static linking: tell the linker to just search for *libfoo.a* when passed the option `-lfoo`.

`-b` *target* *(gcc)*

Cross-compile for machine *target*.

`-C` *(gcc, SGI ANSI C, SCO UNIX, SunOS, SVR4)*

Tell the preprocessor not to discard comments. Used with the `-E` option.

`-c` *(all)*

Stop compiler after producing the object file, do not link.

`-call_shared` *(older MIPS)*

Produce an executable that uses sharable objects (default). On more modern SGI machines, this is called `-KPIC`.

`-cckr` *(SGI)*

Define K&R-style preprocessor variables.

`-common` (SGI)

Cause multiple definitions of global data to be considered to be one definition, and do not produce error messages.

`-compat` (SCO UNIX)

Create an exectauble which is binary compatible across a number of Intel-based systems. Use XENIX libraries to link.

`-cord` *(SGI)*

Rearrange functions in the object file to reduce cache conflicts.

`-CSON` *(SCO UNIX)*

Enable common subexpression optimization. Used in conjunction with the `-O`option.

`-CSOFF` *(SCO UNIX)*

Disable common subexpression optimization. Used in conjunction with the `-O`option.

`-D` *(all)*

Define a preprocessor macro. The form `-D`*foo* defines *foo*, but does not give it a value. This can be tested with *#ifdef* and friends. The form `-Dfoo=3` defines *foo* to have the value 3. This can be tested with *#if*.

`-d` *(XENIX, SCO UNIX)*

Report the compiler passes and arguments as they are executed.

`-d` *when* *(gcc)*

Make dumps during compilation for debugging the compiler. *when* specifies when the dump should be made. Most of these should not be needed by normal users, however the forms `-dD` (leave all macro definitions in the preprocessor output), `-dM` (dump only the macro definitions in effect at the end of preprocessing) and `-dN` (like `-dD` except that only the macro names are output) can be used with the `-E` option in order to debug preprocessor macros.

`-dalign` *(SunOS on Sun-4 systems)*

Generate double load/store instructions for better performance

`-dD` *(gcc)*

Special case of the `-d` option: leave all macro definitions in the preprocessor output. The resulting output will probably not compile, but it's useful for debugging preprocessor macros.

`-dl` *(SVR3)*

Don't generate line number information for the symbolic debugger.

`-dM` *(gcc)*

Special case of the `-d` option: dump only the macro definitions in effect at the end of preprocessing.

`-dn` *(SVR4)*

Don't use dynamic linking. This cannot be used with the `-G` option.

`-dollar` *(SGI)*

Allow the symbol `$` in C identifiers.

`-dos` *(SCO UNIX, XENIX)*

Create an executable for MS-DOS systems.

`-dryrun` *(SunOS)*

Display the commands that the compiler would execute, but do not execute them

`-ds` *(SVR3)*

Don't generate symbol attribute information for the symbolic debugger. This option and `-dl` can be combined as `-dsl`. Together they are the opposite of the `-g` option.

`-dy` *(SVR4)*

Use dynamic linking where possible. This is the default.

`-E` *(all)*

Write preprocessor output to standard output, then stop. Some compilers interpret the `-o` option and write the output there instead if specified.

`-EP` *(SCO UNIX, XENIX)*

Use this instead of the `-E` option to generate preprocessor output without *#line* directives. The output is written to standard output. In addition, SCO UNIX copies the output to a file with the suffix *.i*.

`-F` *num* *(SCO UNIX, XENIX)*

Set the size of the program stack to *num* (hexadecimal) bytes.

`-f` *(gcc)*

A family of options specifying details of C dialect to be compiled. See page 405 for more details.

`-f` *type* *(SunOS)*

Specify the kind of floating-point code to generate on Sun-2, Sun-3 and Sun-4 systems.

`-Fa` *name* *(SCO UNIX, XENIX)*

Write an assembler source listing to *name* (default ***file.s***).

`-Fc` *name* *(SCO UNIX, XENIX)*

Write a merged assembler and C source listing to *name* (default ***file.L***).

`-feedback` *name* *(SGI)*

Specify the name of the feedback file used in conjunction with the `-cord` option.

`-Fe` *name* *(SCO UNIX, XENIX)*

Specify the name of the executable file.

`-Fl` *name* *(SCO UNIX, XENIX)*

Write an assembler listing with assembler source and object code to *name* (default ***file.L***).

`-float` *(SGI)*

Cause the compiler not to promote `float` to `double`.

`-Fm name` *(SCO UNIX, XENIX)*

Write a load map to `name` (default *a.map*).

`-Fo` *name* *(SCO UNIX, XENIX)*

Specify the name of the object file.

`-Fp` *(SCO UNIX, XENIX)*

Specify floating point arithmetic options for MS-DOS cross-compilation.

`-framepointer` *(SGI)*

Use a register other than the stack pointer (sp) for the frame pointers (see Chapter 21, *Object files and friends*, page 377).

`-fullwarn` *(SGI)*

Produce all possible warnings.

`-Fs` *name* *(SCO UNIX, XENIX)*

Write a C source listing to *name* (default *file.**S***).

`-G` *(SVR4)*

Instruct the linker to create a shared object rather than a dynamically linked executable. This is incompatible with the `-dn` option.

`-G` *size* *(SGI)*

Limit items to be placed in the global pointer area to *size* bytes.

`-g` *(all)*

Create additional symbolic information in order to support symbolic debuggers. *gcc* has a number of suboptions to specify the amount and the nature of the debugging information—see page 406 for more details. SGI C specifies a numeric level for the amount of debug information to produce.

`-Gc` *(SCO UNIX)*

Generate code with the alternate calling sequence and naming conventions used in System V 386 Pascal and System V 386 FORTRAN.

`-go` *(SunOS)*

Produce additional symbol table information for *adb*.

`-Gs` *(SCO UNIX)*

Removes stack probe routines. Effective only in non-protected environments.

`-H` *(gcc, System V)*

Print the names of header files to the standard output as they are *#include*d.

`-H` *num* *(SCO UNIX, XENIX)*

Set the maximum length of external symbols to *num*.

`-help` *(SCO UNIX, SunOS)*

Display help for *cc*.

`-I` *dir* *(all)*

Add *dir* to a list of pathnames to search for header files included by the *#include* directive.

`-I` *(SGI)*

Remove */usr/include* from the list of paths to search for header files.

`-I-` *(gcc)*

Search the list of include pathnames only when the *#include* directive is of the form *#include "header"*. Do not search these directories if the directive is *#include <header>*. In addition, do not search the current directory for header files. If `-I` *dir* options are specified after `-I-`, they apply for all forms of the *#include* directive.

`-i` *(SCO UNIX, XENIX)*

Create separate instruction and data spaces for small model programs.

`-J` *(SCO UNIX)*

Change the default mode for the char type to unsigned.

`-J` *(SunOS, Sun-2 and Sun-3)*

Generate 32-bit offsets in `switch` statements.

`-J` *sfm* *(SVR4)*

Specify the pathname of the assembly language source math library *libsfm.sa*. The positioning of this option is important, since the library is searched when the name is encountered.

`-j` *(SGI)*

Create a file *file.**u*** containing intermediate code. Does not create an object file unless used in conjunction with `-c`.

`-KPIC` *(SGI)*

Generate position-independent code.

`-imacros` *file* *(gcc)*

Process *file* before reading the regular input. Do not produce any output for *file*—only the macro definitions will be of use.

`-include` *file* *(gcc)*

Process *file* as input before processing the regular input file. The text of the file will be handled exactly like the regular files.

`-K` *(SVR4)*

Specify various code generation options.

`-K` *(SCO UNIX, XENIX)*

Remove stack probes from a program. Useful only in non-protected environments.

`-k` *options* *(SGI)*

Pass *options* to the ucode loader.

`-ko` *name* *(SGI)*

Cause the output of the intermediate code loader to be called *name*.

`-L` *(SCO UNIX, XENIX)*

Create an assembler listing with assembled code and assembler source instructions with the name *file.**L***.

`-L` *dir* *(All but SCO UNIX, XENIX)*

Add *dir* to the list of directories to search to resolve library references. See Chapter 18, *Function libraries*, page 369 for further details.

`-l` *(all but XENIX)*

Specify a library. The option `-lbaz` will search the library paths specified via `-L` options (see above) for a file typically called *libbaz.a*. See Chapter 18, *Function libraries*, page 369 for more details.

`-LARGE` *(SCO UNIX, XENIX)*

Invoke the large model compiler to run. Used if heap space problems occur during compilation.

`-link` *specs* *(SCO UNIX, XENIX)*

Pass *specs* to the linker. All text following up to the end of the command line is passed to the linker, so this has to be the last command on the line.

`-M` *(SVR3)*

Instruct the linker to output a message for each multiply defined external symbol.

`-M` *(gcc, SGI, SunOS, Solaris)*

Instruct the preprocessor to write a list of *Makefile* dependencies to *stdout*. Suppress normal preprocessor output.

`-MM` *(gcc)*

Like the `-M` option, but only process *#include "file"* directives—ignore *#include <file>*.

`-MD` *(gcc)*

Like the `-M` directive, but output to a file whose name is made by replacing the final *.c* with *.d*. This option does not suppress preprocessor output.

`-MDupdate` *file* *(SGI)*

While compiling, update *file* to contain header, library and runtime dependency information for the output file.

`-MMD` *(gcc)*

Combination of `-MD` and `-MM`. Does not suppress preprocessor output.

`-Ma` *(SCO UNIX, XENIX)*

Compile strict ANSI.

`-M` *model* *(SCO UNIX, XENIX)*

Select model (only 16-bit modes). *model* may be `c` (compact), `s` (small), `m` (medium), `l` (large) or `h` (huge).

`-M` *num* *(SCO UNIX, XENIX)*

Specify processor model for which code should be generated. `0` specifies 8086, `1` specifies 80186, `2` specifies 80286 and `3` specifies 80386 or later. 16-bit models (`0` to `2`) may be followed by models `s`, `m` or `l`.

`-Mb` *(SCO UNIX, XENIX)*

Reverse the word order for long types.

`-Md` *(SCO UNIX, XENIX)*

Generate code for separate stack and data segments.

`-Me` *(SCO UNIX, XENIX)*

Enable the keywords `far`, `near`, `huge`, `pascal` and `fortran`.

`-Mf` *(SCO UNIX, XENIX)*

Enable software floating point.

`-Mt` *num* *(SCO UNIX, XENIX)*

Set the maximum size of data items to *num*. Only valid for large model.

`-m` *(SVR3)*

Write a load map to standard output.

`-m` *file* *(SCO UNIX, XENIX)*

Write a load map to *file*.

`-mips`*num* *(SGI)*

Specify the target machine. *num* `1` (default) generates code for R2000/R3000, and `2` generates code for R4000.

`-misalign` *(SunOS on Sun-4)*

Generate code to allow loading and storing misaligned data.

`-mp`*(SGI)*

Enable multiprocessing directives.

`-n` *(SCO UNIX, XENIX)*

Select pure text model (separated text and data).

`-ND` *name* *(SCO UNIX, XENIX)*

Set the names of each data segment to *name*.

`-nl` *num* *(SCO UNIX, XENIX)*

Set the maximum length of external symbols to *num*.

`-NM` *name* *(SCO UNIX, XENIX)*

Set the names of each module to *name*.

`-nocpp` *(SGI)*

Do not run the preprocessor when compiling.

`-nointl` *(SCO UNIX)*

Create a binary without international functionality.

`-non_shared` *(SGI)*

Produce an executable that does not use shared objects.

`-noprototypes` *(SGI)*

Remove prototype error and warning messages when run in `-cckr` mode.

`-nostdinc` *(gcc, SGI)*

Do not search the standard include file locations (like */usr/include*) for header files. Only search the directories specified with the `-I` option. *gcc* also has a version `-nostdinc++` for C++ programs.

`-nostdlib` *(gcc)*

Don't include the standard startup files and library paths when linking. Only files explicitly mentioned on the command line will be included.

`-NT` *name* *(SCO UNIX, XENIX)*

Set the names of each text segment to *name*.

`-O` *(all)*

Perform optimizations. In some, it may be followed by a level number (`-O1` normal optimizations, `-O2` additional optimizations, etc.). `-O` means the same thing as `-O1`. Others, such as the SCO compiler, use letters to specify specific optimizations.

`-o` *file* *(all)*

Name the output file *file*. System V compilers only use this option to specify the name of the final executable, whereas other compilers use it to specify the name of the output of the final compiler pass. This can give rise to compatibility problems—see Chapter 20, *Compilers*, page 351 for further details.

`-oldcpp` *(SGI)*

Run with old-style *cpp*.

`-Olimit` *size* *(SGI)*

Set the maximum size of a routine to be optimized by the global optimizer to *size* basic blocks.

`-os2` *(SCO UNIX)*

Create an executable program for OS/2.

`-P` *(gcc)*

Instruct the preprocessor not to generate *#line* commands. Used with the `-E` option.

`-P` *(SunOS, SGI, SVR4, SCO UNIX, XENIX)*

Use instead of the `-E` option to generate preprocessor output without *#line* directives. The output will be stored in *file.**i***.

`-p` *(all)*

Generate extra code to aid profiling using the profiling program *prof*.

`-pack` *(SCO UNIX, XENIX)*

Ignore alignment considerations in structs and pack as tightly as possible.

`-pca` *(SGI)*

Run the *pca* processor to discover parallelism in the source code.

`-pedantic` *(gcc, SGI)*

Be pedantic about syntax checking, issue all required warnings. The variety `-pedantic-errors` treats them as errors instead of warnings.

`-pg` *(gcc, SunOS)*

Like `-p`, except that the output is suitable for processing by the *gprof* profiler.

`-pic`, `-PIC` *(SunOS)*

Generate position-independent code. The form `-PIC` allows a larger global offset table.

`-pipe` *(gcc, SunOS)*

Specify that output from one pass should be piped to the next pass, rather than the more traditional technique of storing it in a temporary file.

`-prototypes` *(SGI)*

Output ANSI function prototypes for all functions in the source file when run in `-cckr` mode.

`-qp` *(System V)*

A synonym for `-p`.

`-Qn` *(gcc (System V versions), SVR4)*

Do not output `.ident` directives to the assembler output to identify the versions of each tool used in the output file.

`-Qy` *(gcc (System V versions), SVR4)*

Output *.ident* directives to the assembler output to identify the versions of each tool used in the output file.

`-Q`*prog opt* *(SunOS)*

Pass option *opt* to program *prog*. *prog* may be `as` (the assembler), `cpp` (the preprocessor), `inline` (the assembly code reorganizer) or `ld` (the loader).

`-Q`*path* *(SunOS)*

Specify search paths for compiler passes and other internal files, such as **crt*.o*.

`-Qproduce` *type* *(SunOS)*

Produce source code output of type *type*. *type* specifies the filename extension and may be one of ***.c*** (C source), ***.i*** (preprocessor output), ***.o*** (object output from the assembler) or ***.s*** (assembler output from the compiler).

`-R` *(SunOS)*

Merge the data segment into text. This creates read-only data.

`-r` *(SCO UNIX, XENIX)*

Invoke the incremental linker */lib/ldr* for the link step.

`-r` *(SVR3)*

Instruct the linker to retain relocation information in the final executable.

`-S` *(gcc, SGI, SunOS, System V)*

Stop after compiling the output assembler code, and do not assemble it. Save the results in a file *file**.s***.

`-S` *(SCO UNIX, XENIX)*

Create a human-readable assembler source listing in *file**.s***. This listing is not suitable for assembly.

`-s` *(SCO UNIX, XENIX, SVR3)*

Strip the final executable.

`-save-temps` *(gcc)*

Keep intermediate files even when they are no longer needed.

`-sb` *(SunOS)*

Generate additional symbol table information for the Sun Source Code Browser.

`-SEG` *num* *(SCO UNIX, XENIX)*

Set the maximum number of segments that the linker can handle to *num*.

`-shared` *(gcc)*

Produce a shared object which can be linked with other objects to form an executable.

`-show` *(SGI)*

Print the names of the passes and their arguments during compilation.

`-signed` *(SGI)*

Use signed characters instead of the default unsigned characters.

`-sopt` *(SGI)*

Invoke the C source-to-source optimizer. There is nothing corresponding to this on other platforms.

`-Ss` *subtitle* *(SCO UNIX)*

Sets subtitle of the source listing. This also causes the linker pass to be omitted.

`-St` *title* *(SCO UNIX)*

Sets title of the source listing. This also causes the linker pass to be omitted.

`-static` *(gcc)*

Produce a statically linked object. This is only of interest on systems which have shared libraries.

`-systype` *(MIPS)*

Specify the name of the compilation environment. Valid names are `bsd4`, `svr3` and `svr4`.

`-t` *(SVR3)*

Instruct the linker to suppress warnings about multiply defined symbols that are not the same size.

`-target` *arch* *(SunOS)*

Specify the target machine. *arch* can be one of `sun2`, `sun3` or `sun4`.

`-Tc` *(SCO UNIX)*

Specify that the input file is a C source file. This can be used if the file does not have a standard *.c* file name extension.

`-temp=`*dir* *(SunOS)*

Store compiler temporary files in *dir*.

`-time` *(SunOS)*

Print time information for each compiler pass.

`-traditional` *(gcc)*

Treat the input sources as pre-ANSI-C. There is also an option `-traditional-cpp` which only affects the preprocessor.

`-trigraphs` *(gcc)*

Enable trigraph processing. By default, trigraphs are disabled unless the `-ansi` option is specified.

`-U` *macro* *(all)*

Undefine *macro*.

`-u` *symbol* *(gcc, SVR3)*

Force the linker to resolve the symbol *symbol* by searching additional libraries where specified.

`-u` *(SCO UNIX)*

Undefine all predefined macros.

`-undef` *(gcc)*

Do not predefine standard macros. This includes the macros which define the architecture.

`-use-readonly-const` *(SGI)*

Do not allow writing to strings and aggregate constants.

`-use-readwrite-const` *(SGI)*

Allow writing to strings and aggregate constants.

`-V` *(System V)*

Print version numbers of the compiler passes as they are invoked.

`-V` *version* *(gcc 2.X)*

Tell *gcc* to run version *version* of *gcc*.

`-V`*"string"* *(SCO UNIX)*

Place *string* in the object file, typically for use as a copyright notice or version information.

`-V` *version* *(XENIX)*

Compile a program compatible with specific versions of UNIX. *version* may be 2 (Seventh Edition compatible), 3 (System III compatible) or 5 (System V compatible).

`-v` *(gcc, SGI)*

Produce verbose output. gcc output includes the complete invocation parameters of each pass and the version numbers of the passes.

`-v` *(SVR4)*

Perform more and stricter semantic checks.

`-varargs` *(SGI)*

Print warnings for lines that may requires the *varargs.h* macros.

`-W` *(gcc)*

Without print a number of additional warning messages. With an argument, add a specific kind of warning message check—see page 407 for more details.

`-W` *num* *(SCO UNIX, XENIX)*

Specify the level of warning messages. If *num* is 0, no warnings are produced. A maximum number of warnings is produced by `-W3`.

`-W0,`*option* *(System V)*

Pass *option* to the compiler.

`-W2,`*option* *(System V)*

Pass *option* to the optimizer.

`-Wa,`*option* *(gcc, System V)*

Pass *option* to the assembler.

`-Wb,`*option* *(System V)*

Pass *option* to the basic block analyzer.

`-Wl,`*option* *(gcc, System V)*

Pass *option* to the linker.

`-Wp,`*option* *(System V)*

Pass *option* to the preprocessor.

`-w` *(gcc, SCO UNIX, SunOS, XENIX)*

Inhibit warning messages.

`-w` *num* *(SGI)*

If *num* is `0` or `1`, suppress warning messages. If *num* is `2`, treat warnings as errors.

`-wline` *(SGI)*

Produce *lint*-like warning messages.

`-woff` *numbers* *(SGI)*

Suppress warning messages corresponding to *numbers*.

`-X` *(SCO UNIX, XENIX)*

Remove the standard directories from the list of directories to searched for *#include* files.

`-Xa` *(SVR4)*

Compile full ANSI C. Extensions are enabled.

`-Xc` *(SVR4)*

Compile strictly conforming ANSI C. Extensions are disabled.

`-Xcpluscomm` *(SGI)*

Allow the C++ comment delimiter `//` when processing C code.

`-xansi` *(SGI)*

Process ANSI C, but accept the extensions allowed by `-cckr`.

`-xenix` *(SCO UNIX)*

Produce XENIX programs using XENIX libraries and include files.

`-xgot` *(SGI)*

Compile using a 32 bit offset in the Global Symbol Table. This can be ignored for other systems.

`-x2.3` *(SCO UNIX)*

Produce XENIX programs using XENIX libraries and include files. The programs are compatible with release 2.3 of XENIX (the last release, with 80386 capabilities).

`-Xlinker,`*option* *(gcc)*

Pass *option* to the linker.

`-Xp` *(SVR3)*

Compile for a POSIX.1 environment.

`-Xs` *(SVR3)*

Compile for a System V.3 environment (i.e. not POSIX.1).

`-Xt` *(SVR4)*

Compile pre-ANSI C, but with compatibility warnings.

`-x` *(SVR3)*

Instruct the linker to save space by not preserving local symbols in the final executable.

`-x` *lang* *(gcc)*

Specify the language to be compiled. *lang* may be one of `c`, `objective-c`, `c-header`, `c++`, `cpp-output`, `assembler` or `assembler-with-cpp`. This overrides the filename extensions.

`-Y0,`*dir* *(SVR3)*

Search for compiler in directory *dir*.

`-Y2,`*dir* *(SVR3)*

Search for optimizer in directory *dir*.

`-Ya,`*dir* *(SVR3)*

Search for assembler in directory *dir*.

`-Yb,`*dir* *(SVR3)*

Search for basic block analyzer in directory *dir*.

`-YI,`*dir* *(SVR3)*

Search for Default include directory in directory *dir*.

`-Yl,`*dir* *(SVR3)*

Search for link editor in directory *dir*.

`-YL,`*dir* *(SVR3)*

Search for first default library directory in directory *dir*.

`-Ym,`*dir* *(gcc (System V versions))*

Search for *m4* in directory *dir*.

`-YP,`*dirs* *(SVR3, gcc (System V versions))*

Tell the compiler to search the directories *dirs* (a colon-separated list, like the PATH environment variable) for libraries specified via the `-l` option. This is an alternative to `-L`. It is not additive: only the directories specified in the last `-YP` option are searched.

`-Yp,`*dir* *(SVR3)*

Search for compiler in directory *dir*.

`-YS,`*dir* *(SVR3)*

Search for startup files *crt1.o* and *crtend.o* in directory *dir*.

`-YU,`*dir* *(SVR3)*

Search for second default library directory in directory *dir*.

`-z` *(SCO UNIX, XENIX)*

Display the passes and arguments, but do not execute them.

`-z` *(SVR3)*

Instruct the linker not to bind anything at address 0 to aid run-time detection of null pointers.

`-Za` *(SCO UNIX, XENIX)*

Restrict the language to ANSI specifications.

`-Zd` *(SCO UNIX, XENIX)*

Include line number information in the object file.

`-Ze` *(SCO UNIX)*

Enables the keywords `far`, `near`, `huge`, `pascal` and `fortran` keywords. The same as the `-Me` option.

`-Zi` *(SCO UNIX, XENIX)*

Include symbolic information in the object file.

`-Zl` *(SCO UNIX)*

Do not include default library information in the object file.

`-Zp`*align* *(SCO UNIX, XENIX, SVR3)*

Force structs to align to the an *align* boundaries. *align* may be `0`, `2` or `4`, and defaults to `1`.

`-Zs` *(SCO UNIX, XENIX)*

Perform syntax check only, do not compile.

gcc dialect options

gcc supplies a large number of options to specify what dialect of C should be compiled. In addition, it supplies a further large number of options for C++ dialect. We'll only look at the C dialect options here—check the *gcc* release for the complete documentation.

`-ansi`

Compile ANSI C. Flag any non-standard extension as warnings, but do not treat them as errors. This option implies the options `-fn-asm` and `-trigraphs`.

`-fno-asm`

Do not recognize the keywords `asm`, `inline` or `typeof`, so that they can be used as

identifiers. The keywords `__asm__`, `__inline__` and `__typeof__` can be used instead.

`-fno-builtin`

Don't recognize builtin function names that do not begin with two leading underscores.

`-trigraphs`

Support ANSI C trigraphs.

`-traditional`

Support pre-ansi dialects. This also implies `-funsigned-bitfields` and `-fwritable-strings`.

`-traditional-cpp`

Provide pre-ANSI style preprocessing. This is implied by `-traditional`.

`-fcond-mismatch`

Allow conditional expressions (such as `a: b? c`) where the second and third arguments have different types.

`-funsigned-char`

By default, characterss are unsigned. This effectively makes the declaration `char` the same thing as `unsigned char`.

`-fsigned-char`

By default, characterss are signed. This effectively makes the declaration `char` the same thing as `signed char`.

`-fsigned-bitfields`

Make bit fields signed by default. This is the default action.

`-funsigned-bitfields`

Make bit fields unsigned by default.

`-fno-signed-bitfields`

Make bit fields unsigned by default.

`-fno-unsigned-bitfields`

Make bit fields signed by default. This is the default action.

`-fwritable-strings`

Allocate strings in the data segment, so that the program can write to them. See Chapter 20, *Compilers*, page 338 for a discussion of this misfeature.

`-fallow-single-precision`

Do not perform operations on single precision floating point values with double precision arithmetic. This is only needed if you specify `-traditional`.

gcc debugging options

`-g` *mods*

Produce standard debugging information. This can be used in conjunction with *gdb*. It sometimes includes information that can confuse other debuggers.

`-ggdb` *mods*

Produce debugging information in the native format (if that is supported), including GDB extensions if at all possible.

`-gstabs` *mods*

Produce debugging information in stabs format without GDB extensions.

`-gcoff` *mods*

Produce debugging information in the *COFF* format used by *sdb* on older System V systems.

`-gxcoff` *mods*

Produce debugging information in the *XCOFF* format used *dbs* on IBM RS/6000 systems.

`-gdwarf` *mods*

Produce debugging information in the *DWARF* format used by *sdb* on most SVR4 systems.

mods are optional and may take the values `+` or the digits `1` to `3`:

- `+` specifies that additional information for *gdb* should be included in the output. This may cause other debuggers to reject the object.
- `1` specifies that only minimal debugging information: include information about function names and external variables, but not about local variables or line numbers.
- `2` (the default): include function names, all variables and line numbers.
- In addition, `3` includes macro definitions. Not all systems support this feature.

gcc warning options

`-W`

Print an number of “standard” extra warning messages. See , starting on page , for a discussion of the individual situations.

`-Wimplicit`

Warn if functions or parameters are declared implicitly (in other words, if the explicit declaration is missing).

`-Wreturn-type`

Warn if a function is defined without a return type (in other words, one that defaults to `int`). Also warn if `return` is used without an argument in a non-`void` function.

`-Wunused`

Warn when local or static variables are not used, and if a statement computes a value which is

not used.

`-Wswitch`

Warn if a `switch` statement has an index of an enumeral type and does not cater for all the possible values of the `enum`, or if a `case` value is specified which does not occur in the `enum`.

`-Wcomment`

Warn if the sequence `/*` is found within a comment. This might mean that a comment end is missing.

`-Wtrigraphs`

Warn if trigraphs are encountered. Only effective if `-ftrigraphs` is also specified.

`-Wformat`

Check the parameters supplied to `printf`, `scanf` and friends to ensure that they agree with the format string.

`-Wchar-subscripts`

Warn if an array subscript has type `char`.

`-Wuninitialized`

Warn if an automatic variable is used before it is initialized. This requires the optimizer to be enabled.

`-Wparentheses`

Warn if parentheses are omitted in assignments in contexts where truth values are expected (for example, `if (a = foo ())`, or when unusual and possibly confusing sequences of nested operators occur without parentheses.

`-Wenum-clash`

Warn if `enum` types are mixed. This is only issued for C++ programs. See Chapter 20, *Compilers*, page 339 for further details.

`-Wtemplate-debugging`

Warn if debugging is not fully available for the platform when using templates in a C++ program.

`-Wall`

Specify all of the warning options above. The FSF considers this a good compromise between accuracy and completeness.

`-fsyntax-only`

Check for syntax errors, but don't compile.

`-pedantic`

Issue all warnings specified by ANSI C. Reject programs which use extensions not defined in the Standard. The Free Software Foundation does not consider this to be a useful option, since ANSI C does not specify warnings for all possible situations. It is included because it is required by the ANSI Standard.

`-pedantic-errors`

The same thing as `-pedantic`, but the warnings are treated as errors.

`-w`

Inhibit all warning messages.

`-Wno-import`

Inhibit warning messages about the use of *#import*.

`-Wtraditional`

Warn about: Macro parameters in strings, functions declared external within a block and then referenced outside the block and `switch` statements with `long` indexes. These are treated differently in ANSI and traditional C.

`-Wshadow`

Warn if a local variable shadows another local variable.

`-Wid-clash-`*len*

Warn whenever two different identifiers match in the first *len* characters. To quote the FSF documentation: *This may help you prepare a program that will compile with certain obsolete, brain-damaged compilers.*

`-Wpointer-arith`

Warn about anything that depends on the "size of" a function type or of `void`. GNU C assigns these types a size of `1`, for convenience in calculations with `void *` pointers and pointers to functions.

`-Wcast-qual`

Warn when a cast removes a type qualifier from a pointer, for example if a `const char *` is cast to a `char *`.

`-Wcast-align`

Warn if a pointer is cast to a type which has an increased alignment requirement. For example, warn if a `char *` is cast to an `int *` on machines where integers require specific alignments.

`-Wwrite-strings`

Give string constants the type `const char []`. This will cause a warning to be generated if a string address is copied into a non-`const char *` pointer.

`-Wconversion`

Warn if the existence of a prototype causes a different type conversion from the default, or if a negative integer constant expression is implicitly converted to an unsigned type.

`-Waggregate-return`

Warn when functions that return structures, unions or arrays are defined or called.

`-Wstrict-prototypes`

Warn if a function is declared or defined without specifying the argument types.

`-Wmissing-prototypes`

Warn if a global function is defined without a previous prototype declaration, even if the definition itself provides the prototype. This warning is intended to help detect missing declarations of global functions in header files.

`-Wredundant-decls`

Warn if anything is declared more than once in the same scope, even in cases where multiple declaration is valid and changes nothing.

`-Wnested-externs`

Warn if an `extern` declaration is encountered within an function.

`-Winline`

Warn if a function was declared as inline, or the C++ option `-finline-functions` was specified, and the function cannot be inlined.

`-Woverloaded-virtual`

C++ only: warn when a derived class function declaration may be an error in defining a virtual function.

`-Werror`

Treat all warnings as errors.

cpp options

`-$` *(gcc)*

Disable the use of the character `$` in identifers. This is passed by *gcc* when the `-ansi` option is specified.

`-A` *(gcc)*

`-A`*question*`(`*answer*`)` asserts that the answer to *question* is *answer*. This can used with the preprocessor conditional *#if #question (answer)*.

`-A-` *(gcc)*

Disable standard assertions. In addition, SVR4 *cc* undefines all standard macros except those beginning with __.

`-B` *(SunOS, Solaris)*

Recognize the C++ comment string `//`.

`-C` *(gcc, SVR3, SunOS, Solaris, XENIX)*

Do not strip comments from the preprocessor output.

`-D`*name* *(gcc, SVR3, SunOS, Solaris, XENIX)*

Define *name* as 1. This is the equivalent to specifying `-D`*name*`=1` to *cc* and *not* the same as `-D`*name*.

`-D`*name*`=`*def* *(gcc, SVR3, SunOS, Solaris, XENIX)*

Define *name*. This is the same as the corresponding *cc* option. This will be overridden by `-U`*name* even if the `-U` option appears earlier on the command line.

`-dM` *(gcc)*

Suppress normal preprocessor output and output *#define* commands for all macros instead. This can also be used with an empty file to show the values of predefined macros.

`-dD` *(gcc)*

Do not strip *#define* commands from the preprocessor output. This can be useful for debugging preprocessor macros.

`-H` *(gcc, SVR3, SunOS, Solaris)*

Print the pathnames of included files on *stderr*.

`-I`*dir* *(gcc, SVR3, SunOS, Solaris)*

Add *dir* to the path to search for *#include* directives.

`-I-` *(gcc)*

Search the list of include pathnames only when the *#include* directive is of the form *#include "header"*. Do not search these directories if the directive is *#include*<*header*>. In addition, do not automatically search the current directory for header files. If `-I` *dir* options are specified after `-I-`, they apply for all forms of the *#include* directive.

`-imacros` *file* *(gcc)*

Process *file* before reading the regular input. Do not produce any output for *file*—only the macro definitions will be of use.

`-include` *file* *(gcc)*

Process *file* as input before processing the regular input file. The text of the file will be handled exactly like the regular files.

`-idirafter` *dir* *(gcc)*

Add *dir* to the *second include path*. The second include path is an include path which is searched when a file isn't found in the standard include path (the one built by the `-I` option).

`-iprefix` *prefix* *(gcc)*

Specify a prefix for the `-iwithprefix` option (see next entry).

`-iwithprefix` *dir* *(gcc)*

Add a the directory *prefix/dir* to the second include path. *prefix* must previously have been set with the *iprefix* command.

`-lang-`*language* *(gcc)*

Specify the source language. `-lang-c++` enables the comment sequence `//`, `-lang-objc` enables the *#import* command, `-lang-objc++` enables both, `-lang-c` disables both.

`-lint` *(gcc)*

Replace *lint* commands such as `/* NOTREACHED */` with the corresponding pragma, e.g. `#pragma lint NOTREACHED`.

`-M` *(gcc, SunOS, Solaris)*

Write a list of *Makefile* dependencies to *stdout*. Suppress normal preprocessor output.

`-MM` *(gcc)*

Like the `-M` option, but only process *#include "file"* directives—ignore *#include <file>*.

`-MD` *(gcc)*

Like the `-M` directive, but output to a file whose name is made by replacing the final *.c* with *.d*. This option does not suppress preprocessor output.

`-MMD` *(gcc)*

Combination of `-MD` and `-MM`. Does not suppress preprocessor output.

`-nostdinc` *(gcc)*

Do not search the standard include file locations (like */usr/include*) for header files. Only search the directories specified with the `-I` option. A version `-nostdinc++` exists for C++ programs.

`-P` *(gcc, SVR3, SunOS, Solaris, XENIX)*

Do not output *#line* directives.

`-p` *(SunOS, Solaris)*

Limit the length of preprocessor directives to 8 characters.

`-pedantic` *(gcc)*

Issue the warnings that ANSI C specifies for specific situations. See Page 399 for more details.

`-pedantic-errors` *(gcc)*

If the situations specified in the ANSI Standard occur, option them as errors rather than warnings.

`-R` *(SunOS, Solaris)*

Allow recursive macros.

`-T` *(SVR3, SunOS, Solaris)*

Limit the length of preprocessor directives to 8 characters. For backward compatibility only.

`-traditional` *(gcc)*

Preprocess in the "traditional" (pre-ANSI) manner.

`-trigraphs` *(gcc)*

Recognize and convert trigraphs.

`-undef` *(SunOS, Solaris)*

Undefine all predefined symbols.

`-U`*name* *(SVR3, SunOS, Solaris, XENIX)*

Remove definition of *name*. This will also override `-D` options placed later on the command line.

`-undef` *(gcc)*

Do not predefine standard macros.

`-Y`*dir* *(SunOS, Solaris)*

Search only directory *dir* for *#include* files.

`-Wall` *(gcc)*

Set both `-Wcomment` and `-Wtrigraphs`.

`-Wcomment` *(gcc)*

Warn if the sequence `/*` is found within a comment. This could imply that a comment end is missing.

`-Wtraditional` *(gcc)*

Warn about macro parameters in strings. These are treated differently in ANSI and traditional C.

`-Wtrigraphs` *(gcc)*

Warn if trigraphs are encountered. Only effective if `-ftrigraphs` is also specified.