

The GNU D Compiler

For GCC version 16.0.0 (pre-release)

(GCC)

David Friedman, Iain Buclaw

Published by the Free Software Foundation
51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA

Copyright © 2006-2025 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

| | |
|---------------------|----|
| Option Index | 63 |
| Keyword Index | 65 |

1 Invoking `gdc`

The `gdc` command is the GNU compiler for the D language and supports many of the same options as `gcc`. See Section “Option Summary” in *Using the GNU Compiler Collection (GCC)*. This manual only documents the options specific to `gdc`.

1.1 Input and Output files

For any given input file, the file name suffix determines what kind of compilation is done. The following kinds of input file names are supported:

`file.d` D source files.
`file.dd` Ddoc source files.
`file.di` D interface files.

You can specify more than one input file on the `gdc` command line, each being compiled separately in the compilation process. If you specify a `-o file` option, all the input files are compiled together, producing a single output file, named `file`. This is allowed even when using `-S` or `-c`.

A D interface file contains only what an import of the module needs, rather than the whole implementation of that module. They can be created by `gdc` from a D source file by using the `-H` option. When the compiler resolves an import declaration, it searches for matching `.di` files first, then for `.d`.

A Ddoc source file contains code in the D macro processor language. It is primarily designed for use in producing user documentation from embedded comments, with a slight affinity towards HTML generation. If a `.d` source file starts with the string `Ddoc` then it is treated as general purpose documentation, not as a D source file.

1.2 Runtime Options

These options affect the runtime behavior of programs compiled with `gdc`.

`-fall-instantiations`

Generate code for all template instantiations. The default template emission strategy is to not generate code for declarations that were either instantiated speculatively, such as from `__traits(compiles, ...)`, or that come from an imported module not being compiled.

`-fno-assert`

Turn off code generation for `assert` contracts.

`-fno-bounds-check`

Turns off array bounds checking for all functions, which can improve performance for code that uses arrays extensively. Note that this can result in unpredictable behavior if the code in question actually does violate array bounds constraints. It is safe to use this option if you are sure that your code never throws a `RangeError`.

- `'c++23'` Sets `__traits(getTargetInfo, "cppStd")` to 202302.
- `-finclude-imports`
Include imported modules in the compilation, as if they were given on the command line. When this option is enabled, all imported modules are compiled except those that are part of libphobos.
- `-fno-invariants`
Turns off code generation for class `invariant` contracts.
- `-fmain` Generates a default `main()` function when compiling. This is useful when unittesting a library, as it enables running the unittests in a library without having to manually define an entry-point function. This option does nothing when `main` is already defined in user code.
- `-fno-moduleinfo`
Turns off generation of the `ModuleInfo` and related functions that would become unreferenced without it, which may allow linking to programs not written in D. Functions that are not be generated include module constructors and destructors (`static this` and `static ~this`), `unittest` code, and DSO registry functions for dynamically linked code.
- `-fonly=filename`
Tells the compiler to parse and run semantic analysis on all modules on the command line, but only generate code for the module specified by *filename*.
- `-fno-postconditions`
Turns off code generation for postcondition `out` contracts.
- `-fno-preconditions`
Turns off code generation for precondition `in` contracts.
- `-fpreview=id`
Turns on an upcoming D language change identified by *id*. The following values are supported:
- `'all'` Turns on all upcoming D language features.
 - `'bitfields'`
Implements bit-fields in D.
 - `'dip1000'` Implements <https://github.com/dlang/DIPs/blob/master/DIPs/other/DIP1000.md> (Scoped pointers).
 - `'dip1008'` Implements <https://github.com/dlang/DIPs/blob/master/DIPs/other/DIP1008.md> (Allow exceptions in `@nogc` code).
 - `'dip1021'` Implements <https://github.com/dlang/DIPs/blob/master/DIPs/accepted/DIP1021.md> (Mutable function arguments).
 - `'dip25'` Implements <https://github.com/dlang/DIPs/blob/master/DIPs/archive/DIP25.md> (Sealed references).
 - `'dtorfields'`
Turns on generation for destructing fields of partially constructed objects.

| | |
|-------------------------------------|--|
| <code>'fieldwise'</code> | Turns on generation of struct equality to use field-wise comparisons. |
| <code>'fixaliasthis'</code> | Implements new lookup rules that check the current scope for <code>alias this</code> before searching in upper scopes. |
| <code>'fiximmutableconv'</code> | Disallows unsound immutable conversions that were formerly incorrectly permitted. |
| <code>'in'</code> | Implements <code>in</code> parameters to mean <code>scope const [ref]</code> and accepts rvalues. |
| <code>'inclusiveincontracts'</code> | Implements <code>in</code> contracts of overridden methods to be a superset of parent contract. |
| <code>'nosharedaccess'</code> | Turns off and disallows all access to shared memory objects. |
| <code>'rvaluerefparam'</code> | Implements rvalue arguments to <code>ref</code> parameters. |
| <code>'systemvariables'</code> | Disables access to variables marked <code>@system</code> from <code>@safe</code> code. |

-frelease

Turns on compiling in release mode, which means not emitting runtime checks for contracts and asserts. Array bounds checking is not done for `@system` and `@trusted` functions, and assertion failures are undefined behavior.

This is equivalent to compiling with the following options:

```
gdc -fno-assert -fbounds-check=safe -fno-invariants \
    -fno-postconditions -fno-preconditions -fno-switch-errors
```

-frevert=

Turns off a D language feature identified by *id*. The following values are supported:

| | |
|---------------------------|--|
| <code>'all'</code> | Turns off all revertable D language features. |
| <code>'dip1000'</code> | Reverts https://github.com/dlang/DIPs/blob/master/DIPs/other/DIP1000.md (Scoped pointers). |
| <code>'dip25'</code> | Reverts https://github.com/dlang/DIPs/blob/master/DIPs/archive/DIP25.md (Sealed references). |
| <code>'dtorfields'</code> | Turns off generation for destructing fields of partially constructed objects. |
| <code>'intpromote'</code> | Turns off C-style integral promotion for unary <code>+</code> , <code>-</code> and <code>~</code> expressions. |

- MQ *target***
Same as **-MT**, but it quotes any characters which are special to **make**.
- MD**
This option is equivalent to **-M -MF *file***. The driver determines *file* by removing any directory components and suffix from the input file, and then adding a **.deps** suffix.
- MMD**
Like **-MD** but does not mention imported modules from the D standard library package directories.
- X**
Output information describing the contents of all source files being compiled in JSON format to a file. The driver determines *file* by removing any directory components and suffix from the input file, and then adding a **.json** suffix.
- Xf *file***
Same as **-X**, but writes all JSON contents to the specified *file*.
- fdoc**
Generates Ddoc documentation and writes it to a file. The compiler determines *file* by removing any directory components and suffix from the input file, and then adding a **.html** suffix.
- fdoc-dir=*dir***
Same as **-fdoc**, but writes documentation to directory *dir*. This option can be used with **-fdoc-file=*file*** to independently set the output file and directory path.
- fdoc-file=*file***
Same as **-fdoc**, but writes documentation to *file*. This option can be used with **-fdoc-dir=*dir*** to independently set the output file and directory path.
- fdoc-inc=*file***
Specify *file* as a Ddoc macro file to be read. Multiple **-fdoc-inc** options can be used, and files are read and processed in the same order.
- fdump-c++-spec=*file***
For D source files, generate corresponding C++ declarations in *file*.
- fdump-c++-spec-verbose**
In conjunction with **-fdump-c++-spec=** above, add comments for ignored declarations in the generated C++ header.
- fsave-mixins=*file***
Generates code expanded from D **mixin** statements and writes the processed sources to *file*. This is useful to debug errors in compilation and provides source for debuggers to show when requested.

1.5 Warnings

Warnings are diagnostic messages that report constructions that are not inherently erroneous but that are risky or suggest there is likely to be a bug in the program. Unless **-Werror** is specified, they do not prevent compilation of the program.

- Wall**
Turns on all warnings messages. Warnings are not a defined part of the D language, and all constructs for which this may generate a warning message are valid code.

- Walloca** This option warns on all uses of "alloca" in the source.
- Walloca-larger-than=*n***
Warn on unbounded uses of `alloca`, and on bounded uses of `alloca` whose bound can be larger than *n* bytes. **-Wno-alloc-larger-than** disables **-Walloca-larger-than** warning and is equivalent to **-Walloca-larger-than=SIZE_MAX** or larger.
- Wno-builtin-declaration-mismatch**
Warn if a built-in function is declared with an incompatible signature.
- Wcast-result**
Warn about casts that will produce a null or zero result. Currently this is only done for casting between an imaginary and non-imaginary data type, or casting between a D and C++ class.
- Wno-deprecated**
Do not warn about usage of deprecated features and symbols with **deprecated** attributes.
- Werror** Turns all warnings into errors.
- Wextra** This enables some extra warning flags that are not enabled by **-Wall**.
 - Waddress**
 - Wcast-result**
 - Wmismatched-special-enum**
 - Wunknown-pragmas**
- Wmismatched-special-enum**
Warn when an enum the compiler recognizes as special is declared with a different size to the built-in type it is representing.
- Wspeculative**
List all error messages from speculative compiles, such as `__traits(compiles, ...)`. This option does not report messages as warnings, and these messages therefore never become errors when the **-Werror** option is also used.
- Wunknown-pragmas**
Warn when a `pragma()` is encountered that is not understood by `gdc`. This differs from **-fignore-unknown-pragmas** where a `pragma` that is part of the D language, but not implemented by the compiler, won't get reported.
- Wno-varargs**
Do not warn upon questionable usage of the macros used to handle variable arguments like `va_start`.
- fno-ignore-unknown-pragmas**
Do not recognize unsupported pragmas. Any `pragma()` encountered that is not part of the D language will result in an error. This option is now deprecated and will be removed in a future release.
- fmax-errors=*n***
Limits the maximum number of error messages to *n*, at which point `gdc` bails out rather than attempting to continue processing the source code. If *n* is 0 (the default), there is no limit on the number of error messages produced.

1.7 Developer Options

This section describes command-line options that are primarily of interest to developers or language tooling.

-fdump-d-original

Output the internal front-end AST after the **semantic3** stage. This option is only useful for debugging the GNU D compiler itself.

-v

Dump information about the compiler language processing stages as the source program is being compiled. This includes listing all modules that are processed through the **parse**, **semantic**, **semantic2**, and **semantic3** stages; all **import** modules and their file paths; and all **function** bodies that are being compiled.


```
@(gcc.attributes.noSanitize ("sanitize_option"))
```

This attribute is a synonym for `@no_sanitize("sanitize_option")`.

```
@(gcc.attributes.optStrategy ("strategy"))
```

This attribute is a synonym for `@optimize("O0")` and `@optimize("Os")`. Sets the optimization strategy for a function. Valid strategies are "none", "optsize", "minsize". The strategies are mutually exclusive.

```
@(gcc.attributes.polly)
```

This attribute is a synonym for `@optimize("loop-parallelize-all", "loop-nest-optimize")`. Only effective when GDC was built with ISL included.

2.1.4 Target-specific Attributes

Many targets have their own target-specific attributes. These are also exposed via the `gcc.attributes` module with use of the generic `@(gcc.attributes.attribute)` UDA function.

See Section 2.1.1 [Attribute Syntax], page 11, for details of the exact syntax for using attributes.

See the function and variable attribute documentation in the GCC manual for more information about what attributes are available on each target.

Examples of using x86-specific target attributes are shown as follows:

```
import gcc.attributes;

@attribute("cdecl")
@attribute("fastcall")
@attribute("ms_abi")
@attribute("sysv_abi")
@attribute("callee_pop_aggregate_return", 1)
@attribute("ms_hook_prologue")
@attribute("naked")
@attribute("regparm", 2)
@attribute("sseregparm")
@attribute("force_align_arg_pointer")
@attribute("stdcall")
@attribute("no_caller_saved_registers")
@attribute("interrupt")
@attribute("indirect_branch", "thunk")
@attribute("function_return", "keep"))
@attribute("nof_check")
@attribute("cf_check")
@attribute("indirect_return")
@attribute("fentry_name", "nop")
@attribute("fentry_section", "__entry_loc")
@attribute("nodirect_extern_access")
```

2.2 Built-in Functions

GCC provides a large number of built-in functions that are made available in GNU D by importing the `gcc.builtins` module. Declarations in this module are automatically created by the compiler. All declarations start with `__builtin_`. Refer to the built-in function documentation in the GCC manual for a full list of functions that are available.

2.2.1 Built-in Types

In addition to built-in functions, the following types are defined in the `gcc.builtins` module.

```

__builtin_clong
    The D equivalent of the target's C long type.

__builtin_clonglong
    The D equivalent of the target's C long long type.

__builtin_culong
    The D equivalent of the target's C unsigned long type.

__builtin_culonglong
    The D equivalent of the target's C unsigned long long type.

__builtin_machine_byte
    Signed unit-sized integer type.

__builtin_machine_int
    Signed word-sized integer type.

__builtin_machine_ubyte
    Unsigned unit-sized integer type.

__builtin_machine_uint
    Unsigned word-sized integer type.

__builtin_pointer_int
    Signed pointer-sized integer type.

__builtin_pointer_uint
    Unsigned pointer-sized integer type.

__builtin_unwind_int
    The D equivalent of the target's C _Unwind_Sword type.

__builtin_unwind_uint
    The D equivalent of the target's C _Unwind_Word type.

__builtin_va_list
    The target's va_list type.

```

2.2.2 Querying Available Built-ins

Not all of the functions are supported, and some target-specific functions may only be available when compiling for a particular ISA. One way of finding out what is exposed by the built-ins module is by generating a D interface file. Assuming you have no file `builtins.d`, the command

```
echo "module gcc.builtins;" > builtins.d; gdc -H -fsyntax-only builtins.d
```

will save all built-in declarations to the file `builtins.di`.

Another way to determine whether a specific built-in is available is by using compile-time reflection.

```
enum X86_HAVE_SSE3 = __traits(compiles, __builtin_ia32_haddps);
```


`float std.math.rounding.floor (float x)` [Function]
`double std.math.rounding.floor (double x)` [Function]
`real std.math.rounding.floor (real x)` [Function]
 Returns the value of *x* rounded downward to the next integer (toward negative infinity).
 This function is evaluated during CTFE as the GCC built-in function `__builtin_floor`.

`real std.math.rounding.round (real x)` [Function]
 Return the value of *x* rounded to the nearest integer. If the fractional part of *x* is exactly 0.5, the return value is rounded away from zero.
 This function is evaluated during CTFE as the GCC built-in function `__builtin_round`.

`real std.math.rounding.trunc (real x)` [Function]
 Returns the integer portion of *x*, dropping the fractional portion.
 This function is evaluated during CTFE as the GCC built-in function `__builtin_trunc`.

`R std.math.traits.copysign (R, X)(R to, X from)` [Template]
 Returns a value composed of *to* with *from*'s sign bit.
 This function is evaluated during CTFE as the GCC built-in function `__builtin_copysign`.

`bool std.math.traits.isFinite (X)(X x)` [Template]
 Returns true if *x* is finite.
 This function is evaluated during CTFE as the GCC built-in function `__builtin_isfinite`.

`bool std.math.traits.isInfinity (X)(X x)` [Template]
 Returns true if *x* is infinite.
 This function is evaluated during CTFE as the GCC built-in function `__builtin_isinf`.

`bool std.math.traits.isNaN (X)(X x)` [Template]
 Returns true if *x* is NaN.
 This function is evaluated during CTFE as the GCC built-in function `__builtin_isnan`.

`float std.math.trigoometry.tan (float x)` [Function]
`double std.math.trigoometry.tan (double x)` [Function]
`real std.math.trigonometry.tan (real x)` [Function]
 Returns tangent of *x*, where *x* is in radians.
 This intrinsic is the same as the GCC built-in function `__builtin_tan`.

MIPS32
MIPS64
MIPS_EABI
MIPS_HardFloat
MIPS_N32
MIPS_N64
MIPS_O32
MIPS_O64
MIPS_SoftFloat
 Versions relating to the MIPS family of processors.

NetBSD Version relating to NetBSD systems.

OpenBSD Version relating to OpenBSD systems.

OSX Version relating to OSX systems.

Posix Version relating to POSIX systems (includes Linux, FreeBSD, OSX, Solaris, etc).

PPC
PPC64
PPC_HardFloat
PPC_SoftFloat
 Versions relating to the PowerPC family of processors.

RISCV32
RISCV64 Versions relating to the RISC-V family of processors.

S390
SystemZ Versions relating to the S/390 and System Z family of processors.

S390X Deprecated; use `SystemZ` instead.

Solaris Versions relating to Solaris systems.

SPARC
SPARC64
SPARC_HardFloat
SPARC_SoftFloat
SPARC_V8Plus
 Versions relating to the SPARC family of processors.

Thumb Deprecated; use `ARM_Thumb` instead.

D_X32
X86
X86_64 Versions relating to the x86-32 and x86-64 family of processors.

Windows
Win32
Win64 Versions relating to Microsoft Windows systems.

2.8 Special Enums

Special `enum` names are used to represent types that do not have an equivalent basic D type. For example, C++ types used by the C++ name mangler.

Special enums are declared opaque, with a base type explicitly set. Unlike regular opaque enums, special enums can be used as any other value type. They have a default `.init` value, as well as other enum properties available (`.min`, `.max`). Special enums can be declared in any module, and will be recognized by the compiler.

```
import gcc.builtins;
enum __c_long : __builtin_clong;
__c_long var = 0x800A;
```

The following identifiers are recognized by GNU D.

```
__c_complex_double
    C _Complex double type.

__c_complex_float
    C _Complex float type.

__c_complex_real
    C _Complex long double type.

__c_long    C++ long type.

__c_longlong
    C++ long long type.

__c_long_double
    C long double type.

__c_ulong
    C++ unsigned long type.

__c_ulonglong
    C++ unsigned long long type.

__c_wchar_t
    C++ wchar_t type.
```

The `core.stdc.config` module declares the following shorthand alias types for convenience: `c_complex_double`, `c_complex_float`, `c_complex_real`, `cpp_long`, `cpp_longlong`, `c_long_double`, `cpp_ulong`, `cpp_ulonglong`.

It may cause undefined behavior at runtime if a special enum is declared with a base type that has a different size to the target C/C++ type it is representing. The GNU D compiler will catch such declarations and emit a warning when the `-Wmismatched-special-enum` option is seen on the command-line.

2.9 Traits

Traits are extensions to the D programming language to enable programs, at compile time, to get at information internal to the compiler. This is also known as compile time reflection.

GNU D implements a `__traits(getTargetInfo)` trait that receives a string key as its argument. The result is an expression describing the requested target information.

```
version (OSX)
```


On x86 targets, all intrinsics are available as functions in the `gcc.builtins` module, and have predictable equivalents.

```
version (DigitalMars)
{
    __simd(XMM.PSLLW, op1, op2);
    __simd_ib(XMM.PSLLW, op1, imm8);
}
version (GNU)
{
    __builtin_ia32_psllw(op1, op2);
    __builtin_ia32_psllwi(op1, imm8);
}
```

TypeInfo-based `va_arg`

The Digital Mars D compiler implements a version of `core.vararg.va_arg` that accepts a run-time `TypeInfo` argument for use when the static type is not known. This function is not implemented by GNU D. It is more portable to use variadic template functions instead.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a. The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source.

The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

- be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
 - C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
 - D. Preserve all the copyright notices of the Document.
 - E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
 - F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
 - G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
 - H. Include an unaltered copy of this License.
 - I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
 - J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
 - K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
 - L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
 - M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
 - N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
 - O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

N

| | |
|-------------------|---|
| nophoboslib | 9 |
| nostdinc | 6 |

S

| | |
|------------------------|---|
| shared-libphobos | 9 |
| static-libphobos | 9 |

V

| | |
|---------|----|
| v | 10 |
|---------|----|

W

| | |
|-------------------------------------|---|
| Wall | 7 |
| Walloca | 7 |
| Walloca-larger-than | 8 |
| Wbuiltin-declaration-mismatch | 8 |
| Wcast-result | 8 |
| Wdeprecated | 8 |
| Werror | 8 |
| Wextra | 8 |

| | |
|--|---|
| Wmismatched-special-enum | 8 |
| Wno-all | 7 |
| Wno-alloca-larger-than | 8 |
| Wno-builtin-declaration-mismatch | 8 |
| Wno-cast-result | 8 |
| Wno-deprecated | 8 |
| Wno-error | 8 |
| Wno-extra | 8 |
| Wno-mismatched-special-enum | 8 |
| Wno-speculative | 8 |
| Wno-unknown-pragmas | 8 |
| Wno-varargs | 8 |
| Wspeculative | 8 |
| Wunknown-pragmas | 8 |
| Wvarargs | 8 |

X

| | |
|----------|---|
| X | 7 |
| Xf | 7 |

