GNAT Metrics User's Guide

Release 27.0w

AdaCore

Apr 28, 2026

CONTENTS

CHAPTER

ONE

THE GNAT METRICS TOOL `GNATMETRIC`

The `gnatmetric` tool is a utility for computing various program metrics. It takes an Ada source file as input and generates a file containing the metrics data as output. Various switches control which metrics are reported.

`gnatmetric` is a project-aware tool, as detailed in https://docs.adacore.com/live/wave/gnat_ugn/html/gnat_ugn/gnat_ugn/gnat_utility_programs.html#tool-specific-packages-in-project-files

The `gnatmetric` command has the form

```
$ gnatmetric [ switches ] { filename }
```

where:

- `switches` specify the metrics to compute and define the destination for the output
- Each `filename` is the name of a source file to process. 'Wildcards' are allowed, and the file name may contain path information. If no `filename` is supplied, then the `switches` list must contain at least one `--files` switch (see *Other gnatmetric Switches*). Including both a `--files` switch and one or more `filename` arguments is permitted.

  Note that it is no longer necessary to specify the Ada language version; `gnatmetric` can process Ada source code written in any version from Ada 83 onward without specifying any language version switch.

The following subsections describe the various switches accepted by `gnatmetric`, organized by category.

1.1 Configuration in GPR file

The project file package that can specify `gnatmetric` switches is named `Metrics`.

It supports setting the `Default_Switches ("Ada")` attribute for modifying the tool command-line default arguments for the project.

Note: Switches additionaly provided on the command-line may override these default arguments.

```
package Metrics is
   for Default_Switches ("Ada") use
     ("--generate-xml-output", --  Generate an XML output file
      " --xml-file-name", XML_File_Name, -- Set XML file name
      "--lines-all");  --  Report all the line metrics
end Metrics;
```

The GPR file will be automatically used by `gnatmetric` as a set of default switches (See all the possible switches below).

You can also specify external variables with the `-X` switch

```
gnatmetric -P prj.gpr -XVariable="Value..."
```

This allows you to have a set of predefined GNATmetric presets to choose from using scenario variables computed from the external variables in your GPR file, or to have a default GNATmetric configuration when running it through GNATcheck (see https://docs.adacore.com/live/wave/lkql/html/gnatcheck_rm/generated/predefined_rules.html#metrics-related-rules).

1.2 Output File Control

`gnatmetric` has two output formats. It can generate a textual (human-readable) form, and also XML. By default only textual output is generated.

When generating the output in textual form, `gnatmetric` creates for each Ada source file a corresponding text file containing the computed metrics, except for the case when the set of metrics specified by gnatmetric parameters consists only of metrics that are computed for the whole set of analyzed sources, but not for each Ada source. By default, the name of the file containing metric information for a source is obtained by appending the `.metrix` suffix to the name of the input source file. If not otherwise specified and no project file is specified as `gnatmetric` option this file is placed in the same directory as where the source file is located. If `gnatmetric` has a project file as its parameter, it places all the generated files in the object directory of the project (or in the project source directory if the project does not define an object directory). If `--subdirs` option is specified, the files are placed in the subrirectory of this directory specified by this option.

All the output information generated in XML format is placed in a single file. By default the name of this file is `metrix.xml`. If not otherwise specified and if no project file is specified as `gnatmetric` option this file is placed in the current directory.

Some of the computed metrics are summed over the units passed to `gnatmetric`; for example, the total number of lines of code. By default this information is sent to `stdout`, but a file can be specified with the `--global-file-name` switch.

The following switches control the `gnatmetric` output:

`--generate-xml-output`
Generate XML output.

`--generate-xml-schema`
Generate XML output and an XML schema file that describes the structure of the XML metric report. This schema is assigned to the XML file. The schema file has the same name as the XML output file with `.xml` suffix replaced with `.xsd`.

`--no-text-output`
Do not generate the output in text form (implies `-x`).

`--output-dir=`*`output_dir`*
Put text files with detailed metrics into `output_dir`.

`--output-suffix=`*`file_suffix`*
Use `file_suffix`, instead of `.metrix` in the name of the output file.

`--global-file-name=`*`file_name`*
Put global metrics into `file_name`.

`--xml-file-name=`*`file_name`*
Put the XML output into `file_name` (also implies `--generate-xml-output`).

`--short-file-names`

Use 'short' source file names in the output. (The `gnatmetric` output includes the name(s) of the Ada source file(s) from which the metrics are computed. By default each name includes the absolute path. The `--short-file-names` switch causes `gnatmetric` to exclude all directory information from the file names that are output.)

`--wide-character-encoding=e`
Specify the wide character encoding method for the input and output files. `e` is one of the following:

- *8* - UTF-8 encoding
- *b* - Brackets encoding (default value)

1.3 Disable Metrics For Local Units

`gnatmetric` relies on the GNAT compilation model – one compilation unit per one source file. It computes line metrics for the whole source file, and it also computes syntax and complexity metrics for the file's outermost unit.

By default, `gnatmetric` will also compute all metrics for certain kinds of locally declared program units:

- subprogram (and generic subprogram) bodies;
- package (and generic package) specs and bodies;
- task object and type specifications and bodies;
- protected object and type specifications and bodies.

These kinds of entities will be referred to as *eligible local program units*, or simply *eligible local units*, in the discussion below.

Note that a subprogram declaration, generic instantiation, or renaming declaration only receives metrics computation when it appear as the outermost entity in a source file.

Suppression of metrics computation for eligible local units can be obtained via the following switch:

`--no-local-metrics`
Do not compute detailed metrics for eligible local program units.

1.4 Specifying a set of metrics to compute

By default all the metrics are reported. The switches described in this subsection allow you to control, on an individual basis, whether metrics are reported. If at least one positive metric switch is specified (that is, a switch that defines that a given metric or set of metrics is to be computed), then only explicitly specified metrics are reported.

1.4.1 Line Metrics Control

For each source file, and for each of its eligible local program units, `gnatmetric` computes the following metrics:

- the total number of lines;
- the total number of code lines (i.e., non-blank lines that are not comments)
- the number of comment lines
- the number of code lines containing end-of-line comments;
- the comment percentage: the ratio between the number of lines that contain comments and the number of all non-blank lines, expressed as a percentage

- the number of empty lines and lines containing only space characters and/or format effectors (blank lines)
- the average number of code lines in subprogram bodies, task bodies, entry bodies and statement sequences in package bodies

`gnatmetric` sums the values of the line metrics for all the files being processed and then generates the cumulative results. The tool also computes for all the files being processed the average number of code lines in bodies.

You can use the following switches to select the specific line metrics to be reported.

`--lines-all`
Report all the line metrics

`--no-lines-all`
Do not report any of line metrics

`--lines`
Report the number of all lines

`--no-lines`
Do not report the number of all lines

`--lines-code`
Report the number of code lines

`--no-lines-code`
Do not report the number of code lines

`--lines-comment`
Report the number of comment lines

`--no-lines-comment`
Do not report the number of comment lines

`--lines-eol-comment`
Report the number of code lines containing end-of-line comments

`--no-lines-eol-comment`
Do not report the number of code lines containing end-of-line comments

`--lines-ratio`
Report the comment percentage in the program text

`--no-lines-ratio`
Do not report the comment percentage in the program text

`--lines-blank`
Report the number of blank lines

`--no-lines-blank`
Do not report the number of blank lines

`--lines-average`
Report the average number of code lines in subprogram bodies, task bodies, entry bodies and statement sequences in package bodies.

`--no-lines-average`
Do not report the average number of code lines in subprogram bodies, task bodies, entry bodies and statement sequences in package bodies.

`--lines-spark`
Report the number of lines written in SPARK.

`--no-lines-spark`
Do not report the number of lines written in SPARK.

1.4.2 Syntax Metrics Control

`gnatmetric` computes various syntactic metrics for the outermost unit and for each eligible local unit:

- ***LSLOC ('Logical Source Lines Of Code')***
  The total number of declarations and the total number of statements. Note that the definition of declarations is the one given in the reference manual:

  > "Each of the following is defined to be a declaration: any basic_declaration; an enumeration_literal_specification; a discriminant_specification; a component_declaration; a loop_parameter_specification; a parameter_specification; a subprogram_body; an entry_declaration; an entry_index_specification; a choice_parameter_specification; a generic_formal_parameter_declaration."

  This means for example that each enumeration literal adds one to the count, as well as each subprogram parameter.

- ***Maximal static nesting level of inner program units***
  According to *Ada Reference Manual*, 10.1(1):

  > "A program unit is either a package, a task unit, a protected unit, a protected entry, a generic unit, or an explicitly declared subprogram other than an enumeration literal."

- ***Maximal nesting level of composite syntactic constructs***
  This corresponds to the notion of the maximum nesting level in the GNAT built-in style checks (see https://docs.adacore.com/live/wave/gnat_ugn/html/gnat_ugn/gnat_ugn/building_executable_programs_with_gnat.html#style-checking).

- ***Number of formal parameters***
  Number of formal parameters of a subprogram; if a subprogram does have parameters, then numbers of "in", "out" and "in out" parameters are also reported. This metric is reported for subprogram specifications and for subprogram instantiations. For subprogram bodies, expression functions and null procedures this metric is reported if the construct acts as a subprogram declaration but is not a completion of previous declaration. This metric is not reported for generic and formal subprograms.

For the outermost unit in the file, `gnatmetric` additionally computes the following metrics:

- ***Public subprograms***
  This metric is computed for package specs. It is the number of subprograms and generic subprograms declared in the visible part (including the visible part of nested packages, protected objects, and protected types).

- ***All subprograms***
  This metric is computed for bodies and subunits. The metric is equal to a total number of subprogram bodies in the compilation unit. Neither generic instantiations nor renamings-as-a-body nor body stubs are counted. Any subprogram body is counted, independently of its nesting level and enclosing constructs. Generic bodies and bodies of protected subprograms are counted in the same way as 'usual' subprogram bodies.

- ***Public types***
  This metric is computed for package specs and generic package declarations. It is the total number of types that can be referenced from outside this compilation unit, plus the number of types from all the visible parts of all the visible generic packages. Generic formal types are not counted. Only types, not subtypes, are included.

  Along with the total number of public types, the following types are counted and reported separately:

  - *Abstract types*
  - *Root tagged types^ (abstract, non-abstract, private, non-private). Type extensions are *not* counted*
  - *Private types* (including private extensions)
  - *Task types*
  - *Protected types*

- ***All types***
  This metric is computed for any compilation unit. It is equal to the total number of the declarations of different types given in the compilation unit. The private and the corresponding full type declaration are counted as one type declaration. Incomplete type declarations and generic formal types are not counted. No distinction is made among different kinds of types (abstract, private etc.); the total number of types is reported.

By default, all the syntax metrics are reported. You can use the following switches to select specific syntax metrics.

`--syntax-all`
  Report all the syntax metrics

`--no-syntax-all`
  Do not report any of syntax metrics

`--declarations`
  Report the total number of declarations

`--no-declarations`
  Do not report the total number of declarations

`--statements`
  Report the total number of statements

`--no-statements`
  Do not report the total number of statements

`--public-subprograms`
  Report the number of public subprograms in a compilation unit

`--no-public-subprograms`
  Do not report the number of public subprograms in a compilation unit

`--all-subprograms`
  Report the number of all the subprograms in a compilation unit

`--no-all-subprograms`
  Do not report the number of all the subprograms in a compilation unit

`--public-types`
  Report the number of public types in a compilation unit

`--no-public-types`
  Do not report the number of public types in a compilation unit

`--all-types`
  Report the number of all the types in a compilation unit

`--no-all-types`
  Do not report the number of all the types in a compilation unit

`--unit-nesting`
  Report the maximal program unit nesting level

`--no-unit-nesting`
Do not report the maximal program unit nesting level

`--construct-nesting`
Report the maximal construct nesting level

`--no-construct-nesting`
Do not report the maximal construct nesting level

`--param-number`
Report the number of subprogram parameters

`--no-param-number`
Do not report the number of subprogram parameters

1.4.3 Contract Metrics Control

`--contract-all`
Report all the contract metrics

`--no-contract-all`
Do not report any of the contract metrics

`--contract`
Report the number of public subprograms with contracts

`--no-contract`
Do not report the number of public subprograms with contracts

`--post`
Report the number of public subprograms with postconditions

`--no-post`
Do not report the number of public subprograms with postconditions

`--contract-complete`
Report the number of public subprograms with complete contracts

`--no-contract-complete`
Do not report the number of public subprograms with complete contracts

`--contract-cyclomatic`
Report the McCabe complexity of public subprograms

`--no-contract-cyclomatic`
Do not report the McCabe complexity of public subprograms

1.4.4 Complexity Metrics Control

For a program unit that is an executable body (a subprogram body (including generic bodies), task body, entry body or a package body containing its own statement sequence) `gnatmetric` computes the following complexity metrics:

- McCabe cyclomatic complexity;
- McCabe essential complexity;
- maximal loop nesting level;
- extra exit points (for subprograms);

The McCabe cyclomatic complexity metric is defined in https://www.mccabe.com/pdf/mccabe-nist235r.pdf

According to McCabe, both control statements and short-circuit control forms should be taken into account when computing cyclomatic complexity. For Ada 2012 we have also take into account conditional expressions and quantified expressions. For each body, we compute three metric values:

- the complexity introduced by control statements only, without taking into account short-circuit forms (referred as `statement complexity` in `gnatmetric` output),
- the complexity introduced by short-circuit control forms only (referred as `expression complexity` in `gnatmetric` output), and
- the total cyclomatic complexity, which is the sum of these two values (referred as `cyclomatic complexity` in `gnatmetric` output).

The cyclomatic complexity is also computed for Ada 2012 expression functions. An expression function cannot have statements as its components, so only one metric value is computed as a cyclomatic complexity of an expression function.

The origin of cyclomatic complexity metric is the need to estimate the number of independent paths in the control flow graph that in turn gives the number of tests needed to satisfy paths coverage testing completeness criterion. Considered from the testing point of view, a static Ada `loop` (that is, the `loop` statement having static subtype in loop parameter specification) does not add to cyclomatic complexity. By providing `--no-static-loop` option a user may specify that such loops should not be counted when computing the cyclomatic complexity metric

The Ada essential complexity metric is a McCabe cyclomatic complexity metric counted for the code that is reduced by excluding all the pure structural Ada control statements. An compound statement is considered as a non-structural if it contains a `raise` or `return` statement as it subcomponent, or if it contains a `goto` statement that transfers the control outside the operator. A selective `accept` statement with a `terminate` alternative is considered a non-structural statement. When computing this metric, `exit` statements are treated in the same way as `goto` statements unless the `-ne` option is specified.

The Ada essential complexity metric defined here is intended to quantify the extent to which the software is unstructured. It is adapted from the McCabe essential complexity metric defined in https://www.mccabe.com/pdf/mccabe-nist235r.pdf but is modified to be more suitable for typical Ada usage. For example, short circuit forms are not penalized as unstructured in the Ada essential complexity metric.

When computing cyclomatic and essential complexity, `gnatmetric` skips the code in the exception handlers and in all the nested program units. The code of assertions and predicates (that is, subprogram preconditions and postconditions, subtype predicates and type invariants) is also skipped.

By default, all the complexity metrics are reported. For more fine-grained control you can use the following switches:

`--complexity-all`
Report all the complexity metrics

`--no-complexity-all`
Do not report any of the complexity metrics

`--complexity-cyclomatic`
Report the McCabe Cyclomatic Complexity

`--no-complexity-cyclomatic`
Do not report the McCabe Cyclomatic Complexity

`--complexity-essential`
Report the Essential Complexity

`--no-complexity-essential`
Do not report the Essential Complexity

`--loop-nesting`
Report maximal loop nesting level

`--no-loop-nesting`
Do not report maximal loop nesting level

`--complexity-average`
Report the average McCabe Cyclomatic Complexity for all the subprogram bodies, task bodies, entry bodies and statement sequences in package bodies. The metric is reported for whole set of processed Ada sources only.

`--no-complexity-average`
Do not report the average McCabe Cyclomatic Complexity for all the subprogram bodies, task bodies, entry bodies and statement sequences in package bodies

`--no-treat-exit-as-goto`
Do not consider `exit` statements as `goto`s when computing Essential Complexity

`--no-static-loop`
Do not consider static loops when computing cyclomatic complexity

`--extra-exit-points`
Report the extra exit points for subprogram bodies. As an exit point, this metric counts `return` statements and raise statements in case when the raised exception is not handled in the same body. In case of a function this metric subtracts 1 from the number of exit points, because a function body must contain at least one `return` statement.

`--no-extra-exit-points`
Do not report the extra exit points for subprogram bodies

1.4.5 Coupling Metrics Control

Coupling metrics measure the dependencies between a given entity and other entities in the program. This information is useful since high coupling may signal potential issues with maintainability as the program evolves.

`gnatmetric` computes the following coupling metrics:

- *object-oriented coupling*, for classes in traditional object-oriented sense;
- *unit coupling*, for all the program units making up a program;
- *control coupling*, reflecting dependencies between a unit and other units that contain subprograms.

Two kinds of coupling metrics are computed:

- fan-out coupling ('efferent coupling'): the number of entities the given entity depends upon. This metric reflects how the given entity depends on the changes in the 'external world'.
- fan-in coupling ('afferent' coupling): the number of entities that depend on a given entity. This metric reflects how the 'external world' depends on the changes in a given entity.

Object-oriented coupling metrics measure the dependencies between a given class (or a group of classes) and the other classes in the program. In this subsection the term 'class' is used in its traditional object-oriented programming sense (an instantiable module that contains data and/or method members). A *category* (of classes) is a group of closely related classes that are reused and/or modified together.

A class `K`'s fan-out coupling is the number of classes that `K` depends upon. A category's fan-out coupling is the number of classes outside the category that the classes inside the category depend upon.

A class `K`'s fan-in coupling is the number of classes that depend upon `K`. A category's fan-in coupling is the number of classes outside the category that depend on classes belonging to the category.

Ada's object-oriented paradigm separates the instantiable entity (type) from the module (package), so the definition of the coupling metrics for Ada maps the class and class category notions onto Ada constructs.

For the coupling metrics, several kinds of modules that define a tagged type or an interface type – library packages, library generic packages, and library generic package instantiations – are considered to be classes. A category consists of a library package (or a library generic package) that defines a tagged or an interface type, together with all its descendant (generic) packages that define tagged or interface types. Thus a category is an Ada hierarchy of library-level program units. Class coupling in Ada is referred to as 'tagged coupling', and category coupling is referred to as 'hierarchy coupling'.

For any package serving as a class, its body and subunits (if any) are considered together with its spec when computing dependencies, and coupling metrics are reported for spec units only. Dependencies between classes mean Ada semantic dependencies. For object-oriented coupling metrics, only dependencies on units treated as classes are considered.

Similarly, for unit and control coupling an entity is considered to be the conceptual construct consisting of the entity's specification, body, and any subunits (transitively). `gnatmetric` computes the dependencies of all these units as a whole, but metrics are only reported for spec units (or for a subprogram body unit in case if there is no separate spec for the given subprogram).

For unit coupling, dependencies are computed between all kinds of program units. For control coupling, the dependencies of a given unit are limited to those units that define subprograms. Thus control fan-out coupling is reported for all units, but control fan-in coupling is only reported for units that define subprograms.

The following simple example illustrates the difference between unit coupling and control coupling metrics:

```
package Lib_1 is
    function F_1 (I : Integer) return Integer;
end Lib_1;

package Lib_2 is
    type T_2 is new Integer;
end Lib_2;

package body Lib_1 is
    function F_1 (I : Integer) return Integer is
    begin
       return I + 1;
    end F_1;
end Lib_1;

with Lib_2; use Lib_2;
package Pack is
    Var : T_2;
    function Fun (I : Integer) return Integer;
end Pack;

with Lib_1; use Lib_1;
package body Pack is
    function Fun (I : Integer) return Integer is
    begin
       return F_1 (I);
    end Fun;
end Pack;
```

If we apply `gnatmetric` with the `--coupling-all` option to these units, the result will be:

```
Coupling metrics:
=================
    Unit Lib_1 (C:\\customers\\662\\L406-007\\lib_1.ads)
       control fan-out coupling  : 0
       control fan-in coupling   : 1
       unit fan-out coupling     : 0
       unit fan-in coupling      : 1

    Unit Pack (C:\\customers\\662\\L406-007\\pack.ads)
       control fan-out coupling  : 1
       control fan-in coupling   : 0
       unit fan-out coupling     : 2
       unit fan-in coupling      : 0

    Unit Lib_2 (C:\\customers\\662\\L406-007\\lib_2.ads)
       control fan-out coupling  : 0
       unit fan-out coupling     : 0
       unit fan-in coupling      : 1
```

The result does not contain values for object-oriented coupling because none of the argument units contains a tagged type and therefore none of these units can be treated as a class.

The `Pack` package (spec and body) depends on two units – `Lib_1` and `Lib_2` – and so its unit fan-out coupling is 2. Since nothing depends on it, its unit fan-in coupling is 0, as is its control fan-in coupling. Only one of the units `Pack` depends upon defines a subprogram, so its control fan-out coupling is 1.

`Lib_2` depends on nothing, so its fan-out metrics are 0. It does not define any subprograms, so it has no control fan-in metric. One unit (`Pack`) depends on it , so its unit fan-in coupling is 1.

`Lib_1` is similar to `Lib_2`, but it does define a subprogram. Its control fan-in coupling is 1 (because there is one unit depending on it).

When computing coupling metrics, `gnatmetric` counts only dependencies between units that are arguments of the `gnatmetric` invocation. Coupling metrics are program-wide (or project-wide) metrics, so you should invoke `gnatmetric` for the complete set of sources comprising your program. This can be done by invoking `gnatmetric` with the corresponding project file and with the `-U` option.

By default, all the coupling metrics are reported. You can use the following switches to select specific syntax metrics.

`--coupling-all`
Report all the coupling metrics

`--tagged-coupling-out`
Report tagged (class) fan-out coupling

`--tagged-coupling-in`
Report tagged (class) fan-in coupling

`--hierarchy-coupling-out`
Report hierarchy (category) fan-out coupling

`--hierarchy-coupling-in`
Report hierarchy (category) fan-in coupling

`--unit-coupling-out`
Report unit fan-out coupling

`--unit-coupling-in`
Report unit fan-in coupling

--control-coupling-out
Report control fan-out coupling

--control-coupling-in
Report control fan-in coupling

1.5 Other gnatmetric Switches

Additional gnatmetric switches are as follows:

--version
Display copyright and version, then exit disregarding all other options.

--help
Display usage, then exit disregarding all other options.

-P *file*
Indicates the name of the project file that describes the set of sources to be processed. The exact set of argument sources depends on other options specified, see below. An aggregate project is allowed as the file parameter only if it has exactly one non-aggregate project being aggregated.

-U
If a project file is specified and no argument source is explicitly specified (either directly or by means of -files option), process all the units of the closure of the argument project. Otherwise this option has no effect.

-U *main_unit*
If a project file is specified and no argument source is explicitly specified (either directly or by means of -files option), process the closure of units rooted at main_unit. Otherwise this option has no effect.

-X*name*=*value*
Give external variable name the value value in the argument project. Has no effect if no project is specified.

--RTS=*rts-path*
Specifies the default location of the runtime library. Same meaning as the equivalent gnatmake flag (see https://docs.adacore.com/live/wave/gnat_ugn/html/gnat_ugn/gnat_ugn/building_executable_programs_with_gnat.html#running-gnatmake).

--subdirs=*dir*
Use the specified subdirectory of the project objects file (or of the project file directory if the project does not specify an object directory) for tool output files. Has no effect if no project is specified as tool argument r if --no-objects-dir is specified.

--files=*file*
Take as arguments the files listed in text file file. Text file file may contain empty lines that are ignored. Each nonempty line should contain the name of an existing file. Several such switches may be specified simultaneously.

--ignore=*filename*
Do not process the sources listed in a specified file.

--verbose
Verbose mode; gnatmetric generates version information and then a trace of sources being processed.

--quiet
Quiet mode.

If a project file is specified and no argument source is explicitly specified (either directly or by means of -files option), and no -U is specified, then the set of processed sources is all the immediate units of the argument project.

1.5.1 Legacy Switches

Some switches have a short form, mostly for legacy reasons, as shown below.

-x
    --generate-xml-output

-xs
    --generate-xml-schema

-nt
    --no-text-output

-d *output-dir*
    --output-dir

-o *file-suffix*
    --output-suffix

-og *file-name*
    --global-file-name

-ox *file-name*
    --xml-file-name

-sfn
    --short-file-names

-We
    --wide-character-encoding=*e*

-nolocal
    --no-local-metrics

-ne
    --no-treat-exit-as-goto

-files *filename*
    --files

-v
    --verbose

-q
    --quiet

CHAPTER

TWO

GNU FREE DOCUMENTATION LICENSE

Version 1.3, 3 November 2008

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **Document**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **Opaque**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

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 titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

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 © 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.

INDEX

Symbols

A

C

D

E

F

G

L

M

O

S