Ebrowse

You can browse C++ class hierarchies from within Emacs by using Ebrowse.

This file documents Ebrowse, a C++ class browser for GNU Emacs.

Next: Generating browser files, Previous: Top, Up: Top [Contents][Index]

1 Introduction

When working in software projects using C++, I frequently missed software support for two things:

- When you get a new class library, or you have to work on source code you haven't written yourself (or written sufficiently long ago), you need a tool to let you navigate class hierarchies and investigate features of the software. Without such a tool you often end up `grep`ing through dozens or even hundreds of files.
- Once you are productive, it would be nice to have a tool that knows your sources and can help you while you are editing source code. Imagine to be able to jump to the definition of an identifier while you are editing, or something that can complete long identifier names because it knows what identifiers are defined in your program....

The design of Ebrowse reflects these two needs.

How does it work?

A fast parser written in C is used to process C++ source files. The parser generates a data base containing information about classes, members, global functions, defines, types etc. found in the sources.

The second part of Ebrowse is a Lisp program. This program reads the data base generated by the parser. It displays its contents in various forms and allows you to perform operations on it, or do something with the help of the knowledge contained in the data base.

Navigational use of Ebrowse is centered around two types of buffers which define their own major modes:

Tree buffers are used to view class hierarchies in tree form. They allow you to quickly find classes, find or view class declarations, perform operations like query replace on sets of your source files, and finally tree buffers are used to produce the second buffer form—member buffers. See Tree Buffers.

Members are displayed in *member buffers*. Ebrowse distinguishes between six different types of members; each type is displayed as a member list of its own:

- Instance member variables;
- Instance member functions;
- Static member variables;
- Static member functions;
- Friends/Defines. The list of defines is contained in the friends list of the pseudo-class '`*Globals*`';
- Types (`enum`s, and `typedef`s defined with class scope).

You can switch member buffers from one list to another, or to another class. You

can include inherited members in the display, you can set filters that remove categories of members from the display, and most importantly you can find or view member declarations and definitions with a keystroke. See Member Buffers.

These two buffer types and the commands they provide support the navigational use of the browser. The second form resembles Emacs's Tags package for C and other procedural languages. Ebrowse's commands of this type are not confined to special buffers; they are most often used while you are editing your source code.

To list just a subset of what you can use the Tags part of Ebrowse for:

- Jump to the definition or declaration of an identifier in your source code, with an electric position stack that lets you easily navigate back and forth.
- Complete identifiers in your source with a completion list containing identifiers from your source code only.
- Perform search and query replace operations over some or all of your source files.
- Show all identifiers matching a regular expression—and jump to one of them, if you like.

Next: Loading a Tree, Previous: Overview, Up: Top [Contents][Index]

2 Processing Source Files

Before you can start browsing a class hierarchy, you must run the parser `ebrowse` on your source files in order to generate a Lisp data base describing your program.

The operation of `ebrowse` can be tailored with command line options. Under normal circumstances it suffices to let the parser use its default settings. If you want to do that, call it with a command line like:

```
ebrowse *.h *.cc
```

or, if your shell doesn't allow all the file names to be specified on the command line,

```
ebrowse --files=file
```

where *file* contains the names of the files to be parsed, one per line.

When invoked with option '`--help`', `ebrowse` prints a list of available command line options.

Next: Output file, Up: Generating browser files [Contents][Index]

2.1 Specifying Input Files

'`file`'

Each file name on the command line tells `ebrowse` to parse that file.

'`--files=`*`file`*'

This command line switch specifies that *file* contains a list of file names to parse. Each line in *file* must contain one file name. More than one option of this kind is allowed. You might, for instance, want to use one file for header files, and another for source files.

'`standard input`'

When `ebrowse` finds no file names on the command line, and no '`--file`' option is specified, it reads file names from standard input. This is sometimes convenient when `ebrowse` is used as part of a command pipe.

'`--search-path=`*`paths`*'

This option lets you specify search paths for your input files. *paths* is a list of directory names, separated from each other by a either a colon or a semicolon, depending on the operating system.

It is generally a good idea to specify input files so that header files are parsed before source files. This facilitates the parser's work of properly identifying friend functions of a class.

Next: Structs and unions, Previous: Input files, Up: Generating browser files [Contents][Index]

2.2 Changing the Output File Name

'`--output-file=`*`file`*'

This option instructs `ebrowse` to generate a Lisp data base with name *file*. By default, the data base is named `BROWSE`, and is written in the directory in which `ebrowse` is invoked.

If you regularly use data base names different from the default, you might want to add this to your init file:

```
(add-to-list 'auto-mode-alist '(NAME . ebrowse-tree-mode))
```

where *NAME* is the Lisp data base name you are using.

'`--append`'

By default, each run of `ebrowse` erases the old contents of the output file when writing to it. You can instruct `ebrowse` to append its output to an existing file produced by `ebrowse` with this command line option.

Next: Matching, Previous: Output file, Up: Generating browser files [Contents] [Index]

2.3 Structs and Unions

'`--no-structs-or-unions`'

This switch suppresses all classes in the data base declared as `struct` or `union` in the output.

This is mainly useful when you are converting an existing C program to C++, and do not want to see the old C structs in a class tree.

Next: Verbosity, Previous: Structs and unions, Up: Generating browser files [Contents][Index]

2.4 Regular Expressions

The parser `ebrowse` normally writes regular expressions to its output file that help the Lisp part of Ebrowse to find functions, variables etc. in their source files.

You can instruct `ebrowse` to omit these regular expressions by calling it with the command line switch '`--no-regexps`'.

When you do this, the Lisp part of Ebrowse tries to guess, from member or class names, suitable regular expressions to locate that class or member in source files. This works fine in most cases, but the automatic generation of regular expressions can be too weak if unusual coding styles are used.

'`--no-regexps`'

This option turns off regular expression recording.

'`--min-regexp-length=`*n*'

The number *n* following this option specifies the minimum length of the regular expressions recorded to match class and member declarations and definitions. The default value is set at compilation time of `ebrowse`.

The smaller the minimum length, the higher the probability that Ebrowse will find a wrong match. The larger the value, the larger the output file and therefore the memory consumption once the file is read from Emacs.

'`--max-regexp-length=`*n*'

The number following this option specifies the maximum length of the regular expressions used to match class and member declarations and definitions. The default value is set at compilation time of `ebrowse`.

The larger the maximum length, the higher the probability that the browser will find a correct match, but the larger the value the larger the output file and therefore the memory consumption once the data

is read. As a second effect, the larger the regular expression, the higher the probability that it will no longer match after editing the file.

Previous: Matching, Up: Generating browser files [Contents][Index]

2.5 Verbose Mode

‘--verbose’

When this option is specified on the command line, `ebrowse` prints a period for each file parsed, and it displays a ‘+’ for each class written to the output file.

‘--very-verbose’

This option makes `ebrowse` print out the names of the files and the names of the classes seen.

Next: Tree Buffers, Previous: Generating browser files, Up: Top [Contents] [Index]

3 Starting to Browse

You start browsing a class hierarchy parsed by `ebrowse` by just finding the `BROWSE` file with `C-x C-f`.

An example of a tree buffer display is shown below.

```
|  Collection
|    IndexedCollection
|      Array
|        FixedArray
|    Set
|    Dictionary
```

When you run Emacs on a display which supports colors and the mouse, you will notice that certain areas in the tree buffer are highlighted when you move the mouse over them. This highlight marks mouse-sensitive regions in the buffer. Please notice the help strings in the echo area when the mouse moves over a sensitive region.

A click with `Mouse-3` on a mouse-sensitive region opens a context menu. In addition to this, each buffer also has a buffer-specific menu that is opened with a click with `Mouse-3` somewhere in the buffer where no highlight is displayed.

Next: Member Buffers, Previous: Loading a Tree, Up: Top [Contents][Index]

4 Tree Buffers

Class trees are displayed in *tree buffers* which install their own major mode. Most Emacs keys work in tree buffers in the usual way, e.g., you can move around in the buffer with the usual `C-f`, `C-v` etc., or you can search with `C-s`.

Tree-specific commands are bound to simple keystrokes, similar to `Gnus`. You can take a look at the key bindings by entering `?` which calls `M-x describe-mode` in both tree and member buffers.

Next: Member Display, Up: Tree Buffers [Contents][Index]

4.1 Viewing and Finding Class Declarations

You can view or find a class declaration when the cursor is on a class name.

SPC

This command views the class declaration if the database contains information about it. If you don't parse the entire source you are working on, some classes will only be known to exist but the location of their declarations and definitions will not be known.

RET

Works like SPC, except that it finds the class declaration rather than viewing it, so that it is ready for editing.

The same functionality is available from the menu opened with Mouse-3 on the class name.

Next: Go to Class, Previous: Source Display, Up: Tree Buffers [Contents][Index]

4.2 Displaying Members

Ebrowse distinguishes six different kinds of members, each of which is displayed as a separate *member list*: instance variables, instance functions, static variables, static functions, friend functions, and types.

Each of these lists can be displayed in a member buffer with a command starting with L when the cursor is on a class name. By default, there is only one member buffer named **Members** that is reused each time you display a member list—this has proven to be more practical than to clutter up the buffer list with dozens of member buffers.

If you want to display more than one member list at a time you can *freeze* its member buffer. Freezing a member buffer prevents it from being overwritten the next time you display a member list. You can toggle this buffer status at any time.

Every member list display command in the tree buffer can be used with a prefix argument (C-u). Without a prefix argument, the command will pop to a member buffer displaying the member list. With prefix argument, the member buffer will

additionally be *frozen*.

L v

This command displays the list of instance member variables.

L V

Display the list of static variables.

L d

Display the list of friend functions. This list is used for defines if you are viewing the class '`*Globals*`' which is a place holder for global symbols.

L f

Display the list of member functions.

L F

Display the list of static member functions.

L t

Display a list of types.

These lists are also available from the class' context menu invoked with `Mouse-3` on the class name.

Next: Quitting, Previous: Member Display, Up: Tree Buffers [Contents][Index]

4.3 Finding a Class

/

This command reads a class name from the minibuffer with completion and positions the cursor on the class in the class tree.

If the branch of the class tree containing the class searched for is currently collapsed, the class itself and all its base classes are recursively made visible. (See also Expanding and Collapsing.)

This function is also available from the tree buffer's context menu.

n

Repeat the last search done with `/`. Each tree buffer has its own local copy of the regular expression last searched in it.

Next: File Name Display, Previous: Go to Class, Up: Tree Buffers [Contents] [Index]

4.4 Burying a Tree Buffer

q

Is a synonym for `M-x bury-buffer`.

Next: Expanding and Collapsing, Previous: Quitting, Up: Tree Buffers [Contents] [Index]

4.5 Displaying File Names

T f

This command toggles the display of file names in a tree buffer. If file name display is switched on, the names of the files containing the class declaration are shown to the right of the class names. If the file is not known, the string '`unknown`' is displayed.

This command is also provided in the tree buffer's context menu.

s

Display file names for the current line, or for the number of lines given by a prefix argument.

Here is an example of a tree buffer with file names displayed.

```
|  Collection           (unknown)
|    IndexedCollection  (indexedcltn.h)
|      Array            (array.h)
|        FixedArray     (fixedarray.h)
|    Set                (set.h)
|    Dictionary         (dict.h)
```

Next: Tree Indentation, Previous: File Name Display, Up: Tree Buffers [Contents] [Index]

4.6 Expanding and Collapsing a Tree

You can expand and collapse parts of a tree to reduce the complexity of large class hierarchies. Expanding or collapsing branches of a tree has no impact on the functionality of other commands, like `/`. (See also Go to Class.)

Collapsed branches are indicated with an ellipsis following the class name like in the example below.

```
|  Collection
|    IndexedCollection...
|    Set
|    Dictionary
```

`-`

This command collapses the branch of the tree starting at the class the cursor is on.

`+`

This command expands the branch of the tree starting at the class the cursor is on. Both commands for collapsing and expanding branches are also available from the class' object menu.

`*`

This command expands all collapsed branches in the tree.

Next: Killing Classes, Previous: Expanding and Collapsing, Up: Tree Buffers [Contents][Index]

4.7 Changing the Tree Indentation

T w

This command reads a new indentation width from the minibuffer and redisplays the tree buffer with the new indentation It is also available from the tree buffer's context menu.

Next: Saving a Tree, Previous: Tree Indentation, Up: Tree Buffers [Contents] [Index]

4.8 Removing Classes from the Tree

C-k

This command removes the class the cursor is on and all its derived classes from the tree. The user is asked for confirmation before the deletion is actually performed.

Next: Statistics, Previous: Killing Classes, Up: Tree Buffers [Contents][Index]

4.9 Saving a Tree

C-x C-s

This command writes a class tree to the file from which it was read. This is useful after classes have been deleted from a tree.

C-x C-w

Writes the tree to a file whose name is read from the minibuffer.

Next: Marking Classes, Previous: Saving a Tree, Up: Tree Buffers [Contents] [Index]

4.10 Statistics

x

Display statistics for the tree, like number of classes in it, number of member functions, etc. This command can also be found in the buffer's context menu.

Previous: Statistics, Up: Tree Buffers [Contents][Index]

4.11 Marking Classes

Classes can be marked for operations similar to the standard Emacs commands `M-x tags-search` and `M-x tags-query-replace` (see also See Tags-like Functions.)

M t

Toggle the mark of the line point is in or for as many lines as given by a prefix command. This command can also be found in the class' context menu.

M a

Unmark all classes. With prefix argument `C-u`, mark all classes in the tree. Since this command operates on the whole buffer, it can also be found in the buffer's object menu.

Marked classes are displayed with an `>` in column one of the tree display, like in the following example

```
|> Collection
|    IndexedCollection...
|>   Set
|    Dictionary
```

Next: Tags-like Functions, Previous: Tree Buffers, Up: Top [Contents][Index]

5 Member Buffers

Member buffers are used to operate on lists of members of a class. Ebrowse distinguishes six kinds of lists:

- Instance variables (normal member variables);
- Instance functions (normal member functions);
- Static variables;
- Static member functions;
- Friend functions;
- Types (`enum`s and `typedef`s defined with class scope. Nested classes will be shown in the class tree like normal classes.

Like tree buffers, member buffers install their own major mode. Also like in tree buffers, menus are provided for certain areas in the buffer: members, classes, and the buffer itself.

Next: Finding/Viewing, Up: Member Buffers [Contents][Index]

5.1 Switching Member Lists

L n

This command switches the member buffer display to the next member list.

L p

This command switches the member buffer display to the previous member list.

L f

Switch to the list of member functions.

L F

Switch to the list of static member functions.

L v

Switch to the list of member variables.

L V

Switch to the list of static member variables.

L d

Switch to the list of friends or defines.

L t

Switch to the list of types.

Both commands cycle through the member list.

Most of the commands are also available from the member buffer’s context menu.

Next: Inherited Members, Previous: Switching Member Lists, Up: Member Buffers

[Contents][Index]

5.2 Finding and Viewing Member Source

RET

This command finds the definition of the member the cursor is on. Finding involves roughly the same as the standard Emacs tags facility does—loading the file and searching for a regular expression matching the member.

f

This command finds the declaration of the member the cursor is on.

SPC

This is the same command as `RET`, but views the member definition instead of finding the member's source file.

v

This is the same command as `f`, but views the member's declaration instead of finding the file the declaration is in.

You can install a hook function to perform actions after a member or class declaration or definition has been found, or when it is not found.

All the commands described above can also be found in the context menu displayed when clicking `Mouse-2` on a member name.

Next: Searching Members, Previous: Finding/Viewing, Up: Member Buffers
[Contents][Index]

5.3 Display of Inherited Members

D b

This command toggles the display of inherited members in the member buffer. This is also in the buffer's context menu.

Next: Switching to Tree, Previous: Inherited Members, Up: Member Buffers [Contents][Index]

5.4 Searching Members

G v

Position the cursor on a member whose name is read from the minibuffer; only members shown in the current member buffer appear in the completion list.

G m

Like the above command, but all members for the current class appear in the completion list. If necessary, the current member list is switched to the one containing the member.

With a prefix argument (`C-u`), all members in the class tree, i.e., all members the browser knows about appear in the completion list. The member display will be switched to the class and member list containing the member.

G n

Repeat the last member search.

Look into the buffer's context menu for a convenient way to do this with a mouse.

Next: Filters, Previous: Searching Members, Up: Member Buffers [Contents] [Index]

5.5 Switching to Tree Buffer

TAB

Pop up the tree buffer to which the member buffer belongs.

t

Do the same as `TAB` but also position the cursor on the class displayed in the member buffer.

Next: Attributes, Previous: Switching to Tree, Up: Member Buffers [Contents] [Index]

5.6 Filters

F a u

This command toggles the display of `public` members. The ‘`a`’ stands for ‘access’.

F a o

This command toggles the display of `protected` members.

F a i

This command toggles the display of `private` members.

F v

This command toggles the display of `virtual` members.

F i

This command toggles the display of `inline` members.

F c

This command toggles the display of `const` members.

F p

This command toggles the display of pure virtual members.

F r

This command removes all filters.

These commands are also found in the buffer's context menu.

Next: Long and Short Display, Previous: Filters, Up: Member Buffers [Contents] [Index]

5.7 Displaying Member Attributes

D a

Toggle the display of member attributes (default is on).

The nine member attributes Ebrowse knows about are displayed as a list a single-characters flags enclosed in angle brackets in front the of the member's name. A '`-`' at a given position means that the attribute is false. The list of attributes from left to right is

'`T`'

The member is a template.

'`C`'

The member is declared `extern "C"`.

'`v`'

Means the member is declared `virtual`.

'`i`'

The member is declared `inline`.

'`c`'

The member is `const`.

'`0`'

The member is a pure virtual function.

‘m’

The member is declared `mutable`.

‘e’

The member is declared `explicit`.

‘t’

The member is a function with a throw list.

This command is also in the buffer’s context menu.

Next: Regexp Display, Previous: Attributes, Up: Member Buffers [Contents] [Index]

5.8 Long and Short Member Display

D l

This command toggles the member buffer between short and long display form. The short display form displays member names, only:

```
| isEmpty        contains       hasMember      create
| storeSize      hash           isEqual        restoreGuts
| saveGuts
```

The long display shows one member per line with member name and regular expressions matching the member (if known):

```
| isEmpty               Bool isEmpty () const...
| hash                  unsigned hash () const...
| isEqual               int isEqual (...
```

Regular expressions will only be displayed when the Lisp database has not been produced with the `ebrowse` option ‘`--no-regexps`’. See Regular Expressions.

Next: Switching Classes, Previous: Long and Short Display, Up: Member Buffers

[Contents][Index]

5.9 Display of Regular Expressions

D r

This command toggles the long display form from displaying the regular expressions matching the member declarations to those expressions matching member definitions.

Regular expressions will only be displayed when the Lisp database has not been produced with the `ebrowse` option '`--no-regexps`', see Regular Expressions.

Next: Killing/Burying, Previous: Regexp Display, Up: Member Buffers [Contents] [Index]

5.10 Displaying Another Class

C c

This command lets you switch the member buffer to another class. It reads the name of the new class from the minibuffer with completion.

C b

This is the same command as `C c` but restricts the classes shown in the completion list to immediate base classes, only. If only one base class exists, this one is immediately shown in the minibuffer.

C d

Same as `C b`, but for derived classes.

C p

Switch to the previous class in the class hierarchy on the same level as the class currently displayed.

C n

Switch to the next sibling of the class in the class tree.

Next: Column Width, Previous: Switching Classes, Up: Member Buffers [Contents][Index]

5.11 Burying a Member Buffer

q

This command is a synonym for `M-x bury-buffer`.

Next: Redisplay, Previous: Killing/Burying, Up: Member Buffers [Contents] [Index]

5.12 Setting the Column Width

D w

This command sets the column width depending on the display form used (long or short display).

Next: Getting Help, Previous: Column Width, Up: Member Buffers [Contents] [Index]

5.13 Forced Redisplay

C-l

This command forces a redisplay of the member buffer. If the width of the window displaying the member buffer is changed this command redraws the member list with the appropriate column widths and number of columns.

Previous: Redisplay, Up: Member Buffers [Contents][Index]

5.14 Getting Help

?

This key is bound to `describe-mode`.

Next: GNU Free Documentation License, Previous: Member Buffers, Up: Top [Contents][Index]

6 Tags-like Functions

Ebrowse provides tags functions similar to those of the standard Emacs Tags facility, but better suited to the needs of C++ programmers.

Next: Position Stack, Up: Tags-like Functions [Contents][Index]

6.1 Finding and Viewing Members

The functions in this section are similar to those described in Source Display, and also in Finding/Viewing, except that they work in a C++ source buffer, not in member and tree buffers created by Ebrowse.

C-c C-m f

Find the definition of the member around point. If you invoke this function with a prefix argument, the declaration is searched.

If more than one class contains a member with the given name you

can select the class with completion. If there is a scope declaration in front of the member name, this class name is used as initial input for the completion.

C-c C-m F

Find the declaration of the member around point.

C-c C-m v

View the definition of the member around point.

C-c C-m V

View the declaration of the member around point.

C-c C-m 4 f

Find a member’s definition in another window.

C-c C-m 4 F

Find a member’s declaration in another window.

C-c C-m 4 v

View a member’s definition in another window.

C-c C-m 4 V

View a member’s declaration in another window.

C-c C-m 5 f

Find a member’s definition in another frame.

C-c C-m 5 F

Find a member’s declaration in another frame.

C-c C-m 5 v

View a member’s definition in another frame.

C-c C-m 5 V

View a member’s declaration in another frame.

Next: Search & Replace, Previous: Finding and Viewing, Up: Tags-like Functions [Contents][Index]

6.2 The Position Stack

When jumping to a member declaration or definition with one of Ebrowse’s commands, the position from where you performed the jump and the position where you jumped to are recorded in a *position stack*. There are several ways in which you can quickly move to positions in the stack:

C-c C-m -

This command sets point to the previous position in the position stack. Directly after you performed a jump, this will put you back to the position where you came from.

The stack is not popped, i.e., you can always switch back and forth between positions in the stack. To avoid letting the stack grow to infinite size there is a maximum number of positions defined. When this number is reached, older positions are discarded when new positions are pushed on the stack.

C-c C-m +

This command moves forward in the position stack, setting point to the next position stored in the position stack.

C-c C-m p

Displays an electric buffer showing all positions saved in the stack. You can select a position by pressing SPC in a line. You can view a position with v.

Next: Members in Files, Previous: Position Stack, Up: Tags-like Functions [Contents][Index]

6.3 Searching and Replacing

Ebrowse allows you to perform operations on all or a subset of the files mentioned in a class tree. When you invoke one of the following functions and more than one class tree is loaded, you must choose a class tree to use from an electric tree menu. If the selected tree contains marked classes, the following commands operate on the files mentioned in the marked classes only. Otherwise all files in the class tree are used.

C-c C-m s

This function performs a regular expression search in the chosen set of files.

C-c C-m u

This command performs a search for calls of a given member which is selected in the usual way with completion.

C-c C-m %

Perform a query replace over the set of files.

C-c C-m ,

All three operations above stop when finding a match. You can restart the operation with this command.

C-c C-m n

This restarts the last tags operation with the next file in the list.

Next: Apropos, Previous: Search & Replace, Up: Tags-like Functions [Contents] [Index]

6.4 Members in Files

The command `C-c C-m l`, lists all members in a given file. The file name is read from the minibuffer with completion.

Next: Symbol Completion, Previous: Members in Files, Up: Tags-like Functions [Contents][Index]

6.5 Member Apropos

The command `C-c C-m a` can be used to display all members matching a given regular expression. This command can be very useful if you remember only part of a member name, and not its beginning.

A special buffer is popped up containing all identifiers matching the regular expression, and what kind of symbol it is (e.g., a member function, or a type). You can then switch to this buffer, and use the command `C-c C-m f`, for example, to jump to a specific member.

Next: Member Buffer Display, Previous: Apropos, Up: Tags-like Functions [Contents][Index]

6.6 Symbol Completion

The command `C-c C-m TAB` completes the symbol in front of point.

Previous: Symbol Completion, Up: Tags-like Functions [Contents][Index]

6.7 Quick Member Display

You can quickly display a member buffer containing the member the cursor in on with the command `C-c C-m m`.

Next: Concept Index, Previous: Tags-like Functions, Up: Top [Contents][Index]

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

```
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

1. 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.

2. 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.

3. 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.

4. 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.

5. 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:

1. 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.
2. 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.
3. State on the Title page the name of the publisher of the Modified Version, as the publisher.
4. Preserve all the copyright notices of the Document.
5. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
6. 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.
7. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.
8. Include an unaltered copy of this License.
9. 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.
10. 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.

11. 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.
12. 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.
13. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
14. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
15. 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.

6. 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.”

7. 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.

8. 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.

9. 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.

10. 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.

11. 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 http://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.

12. 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 (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.

Previous: GNU Free Documentation License, Up: Top [Contents][Index]

Concept Index

Jump to:	* - A B C D E F H I K L M N O P R S T U V

Jump to:	* - **A B C D E F H I K L M N O P R S T U V**