GNU History Library

Edition 6.2, for `History Library` Version 6.2.
September 2010

Chet Ramey, Case Western Reserve University
Brian Fox, Free Software Foundation

This document describes the GNU History library (version 6.2, September 6 2010), a programming tool that provides a consistent user interface for recalling lines of previously typed input.

Published by the Free Software Foundation
59 Temple Place, Suite 330,
Boston, MA 02111-1307
USA

Table of Contents

1 Using History Interactively **1**
1.1 History Expansion 1
1.1.1 Event Designators 1
1.1.2 Word Designators 2
1.1.3 Modifiers 2

2 Programming with GNU History **4**
2.1 Introduction to History 4
2.2 History Storage 4
2.3 History Functions 5
2.3.1 Initializing History and State Management 5
2.3.2 History List Management 5
2.3.3 Information About the History List 6
2.3.4 Moving Around the History List 6
2.3.5 Searching the History List 7
2.3.6 Managing the History File 7
2.3.7 History Expansion 8
2.4 History Variables 8
2.5 History Programming Example 10

Appendix A GNU Free Documentation License **12**

Appendix B Concept Index **20**

Appendix C Function and Variable Index **21**

1 Using History Interactively

This chapter describes how to use the GNU History Library interactively, from a user's standpoint. It should be considered a user's guide. For information on using the GNU History Library in your own programs, see Chapter 2 [Programming with GNU History], page 4.

1.1 History Expansion

The History library provides a history expansion feature that is similar to the history expansion provided by `csh`. This section describes the syntax used to manipulate the history information.

History expansions introduce words from the history list into the input stream, making it easy to repeat commands, insert the arguments to a previous command into the current input line, or fix errors in previous commands quickly.

History expansion takes place in two parts. The first is to determine which line from the history list should be used during substitution. The second is to select portions of that line for inclusion into the current one. The line selected from the history is called the *event*, and the portions of that line that are acted upon are called *words*. Various *modifiers* are available to manipulate the selected words. The line is broken into words in the same fashion that Bash does, so that several words surrounded by quotes are considered one word. History expansions are introduced by the appearance of the history expansion character, which is '`!`' by default.

1.1.1 Event Designators

An event designator is a reference to a command line entry in the history list. Unless the reference is absolute, events are relative to the current position in the history list.

`!` Start a history substitution, except when followed by a space, tab, the end of the line, or '`=`'.

`!n` Refer to command line *n*.

`!-n` Refer to the command *n* lines back.

`!!` Refer to the previous command. This is a synonym for '`!-1`'.

`!string` Refer to the most recent command preceding the current position in the history list starting with *string*.

`!?string[?]`
Refer to the most recent command preceding the current position in the history list containing *string*. The trailing '`?`' may be omitted if the *string* is followed immediately by a newline.

`^string1^string2^`
Quick Substitution. Repeat the last command, replacing *string1* with *string2*. Equivalent to `!!:s/string1/string2/`.

`!#` The entire command line typed so far.

1.1.2 Word Designators

Word designators are used to select desired words from the event. A ‘:’ separates the event specification from the word designator. It may be omitted if the word designator begins with a ‘^’, ‘$’, ‘*’, ‘-’, or ‘%’. Words are numbered from the beginning of the line, with the first word being denoted by 0 (zero). Words are inserted into the current line separated by single spaces.

For example,

`!!` designates the preceding command. When you type this, the preceding command is repeated in toto.

`!!:$` designates the last argument of the preceding command. This may be shortened to `!$`.

`!fi:2` designates the second argument of the most recent command starting with the letters `fi`.

Here are the word designators:

`0 (zero)` The `0`th word. For many applications, this is the command word.

n The *n*th word.

`^` The first argument; that is, word 1.

`$` The last argument.

`%` The word matched by the most recent ‘?*string*?’ search.

x-*y* A range of words; ‘-*y*’ abbreviates ‘0-*y*’.

`*` All of the words, except the `0`th. This is a synonym for ‘1-$’. It is not an error to use ‘*’ if there is just one word in the event; the empty string is returned in that case.

*x** Abbreviates ‘*x*-$’

x- Abbreviates ‘*x*-$’ like ‘*x**’, but omits the last word.

If a word designator is supplied without an event specification, the previous command is used as the event.

1.1.3 Modifiers

After the optional word designator, you can add a sequence of one or more of the following modifiers, each preceded by a ‘:’.

`h` Remove a trailing pathname component, leaving only the head.

`t` Remove all leading pathname components, leaving the tail.

`r` Remove a trailing suffix of the form ‘.*suffix*’, leaving the basename.

`e` Remove all but the trailing suffix.

`p` Print the new command but do not execute it.

`s/old/new/`
: Substitute *new* for the first occurrence of *old* in the event line. Any delimiter may be used in place of '`/`'. The delimiter may be quoted in *old* and *new* with a single backslash. If '`&`' appears in *new*, it is replaced by *old*. A single backslash will quote the '`&`'. The final delimiter is optional if it is the last character on the input line.

`&`
: Repeat the previous substitution.

`g`
`a`
: Cause changes to be applied over the entire event line. Used in conjunction with '`s`', as in `gs/old/new/`, or with '`&`'.

`G`
: Apply the following '`s`' modifier once to each word in the event.

2 Programming with GNU History

This chapter describes how to interface programs that you write with the GNU History Library. It should be considered a technical guide. For information on the interactive use of GNU History, see Chapter 1 [Using History Interactively], page 1.

2.1 Introduction to History

Many programs read input from the user a line at a time. The GNU History library is able to keep track of those lines, associate arbitrary data with each line, and utilize information from previous lines in composing new ones.

The programmer using the History library has available functions for remembering lines on a history list, associating arbitrary data with a line, removing lines from the list, searching through the list for a line containing an arbitrary text string, and referencing any line in the list directly. In addition, a history *expansion* function is available which provides for a consistent user interface across different programs.

The user using programs written with the History library has the benefit of a consistent user interface with a set of well-known commands for manipulating the text of previous lines and using that text in new commands. The basic history manipulation commands are similar to the history substitution provided by `csh`.

If the programmer desires, he can use the Readline library, which includes some history manipulation by default, and has the added advantage of command line editing.

Before declaring any functions using any functionality the History library provides in other code, an application writer should include the file `<readline/history.h>` in any file that uses the History library's features. It supplies extern declarations for all of the library's public functions and variables, and declares all of the public data structures.

2.2 History Storage

The history list is an array of history entries. A history entry is declared as follows:

```
typedef void *histdata_t;

typedef struct _hist_entry {
  char *line;
  char *timestamp;
  histdata_t data;
} HIST_ENTRY;
```

The history list itself might therefore be declared as

```
HIST_ENTRY **the_history_list;
```

The state of the History library is encapsulated into a single structure:

```
/*
 * A structure used to pass around the current state of the history.
 */
typedef struct _hist_state {
  HIST_ENTRY **entries; /* Pointer to the entries themselves. */
  int offset;           /* The location pointer within this array. */
```

```
    int length;           /* Number of elements within this array. */
    int size;             /* Number of slots allocated to this array. */
    int flags;
  } HISTORY_STATE;
```

If the flags member includes `HS_STIFLED`, the history has been stifled.

2.3 History Functions

This section describes the calling sequence for the various functions exported by the GNU History library.

2.3.1 Initializing History and State Management

This section describes functions used to initialize and manage the state of the History library when you want to use the history functions in your program.

`void using_history` (*void*) [Function]

Begin a session in which the history functions might be used. This initializes the interactive variables.

`HISTORY_STATE * history_get_history_state` (*void*) [Function]

Return a structure describing the current state of the input history.

`void history_set_history_state` (*HISTORY_STATE *state*) [Function]

Set the state of the history list according to *state*.

2.3.2 History List Management

These functions manage individual entries on the history list, or set parameters managing the list itself.

`void add_history` (*const char *string*) [Function]

Place *string* at the end of the history list. The associated data field (if any) is set to `NULL`.

`void add_history_time` (*const char *string*) [Function]

Change the time stamp associated with the most recent history entry to *string*.

`HIST_ENTRY * remove_history` (*int which*) [Function]

Remove history entry at offset *which* from the history. The removed element is returned so you can free the line, data, and containing structure.

`histdata_t free_history_entry` (*HIST_ENTRY *histent*) [Function]

Free the history entry *histent* and any history library private data associated with it. Returns the application-specific data so the caller can dispose of it.

`HIST_ENTRY * replace_history_entry` (*int which, const char *line, histdata_t data*) [Function]

Make the history entry at offset *which* have *line* and *data*. This returns the old entry so the caller can dispose of any application-specific data. In the case of an invalid *which*, a `NULL` pointer is returned.

`void clear_history` (*void*) [Function]
Clear the history list by deleting all the entries.

`void stifle_history` (*int max*) [Function]
Stifle the history list, remembering only the last *max* entries.

`int unstifle_history` (*void*) [Function]
Stop stifling the history. This returns the previously-set maximum number of history entries (as set by `stifle_history()`). The value is positive if the history was stifled, negative if it wasn't.

`int history_is_stifled` (*void*) [Function]
Returns non-zero if the history is stifled, zero if it is not.

2.3.3 Information About the History List

These functions return information about the entire history list or individual list entries.

`HIST_ENTRY ** history_list` (*void*) [Function]
Return a `NULL` terminated array of `HIST_ENTRY *` which is the current input history. Element 0 of this list is the beginning of time. If there is no history, return `NULL`.

`int where_history` (*void*) [Function]
Returns the offset of the current history element.

`HIST_ENTRY * current_history` (*void*) [Function]
Return the history entry at the current position, as determined by `where_history()`. If there is no entry there, return a `NULL` pointer.

`HIST_ENTRY * history_get` (*int offset*) [Function]
Return the history entry at position *offset*, starting from `history_base` (see Section 2.4 [History Variables], page 8). If there is no entry there, or if *offset* is greater than the history length, return a `NULL` pointer.

`time_t history_get_time` (*HIST_ENTRY *entry*) [Function]
Return the time stamp associated with the history entry *entry*.

`int history_total_bytes` (*void*) [Function]
Return the number of bytes that the primary history entries are using. This function returns the sum of the lengths of all the lines in the history.

2.3.4 Moving Around the History List

These functions allow the current index into the history list to be set or changed.

`int history_set_pos` (*int pos*) [Function]
Set the current history offset to *pos*, an absolute index into the list. Returns 1 on success, 0 if *pos* is less than zero or greater than the number of history entries.

`HIST_ENTRY * previous_history` (*void*) [Function]
Back up the current history offset to the previous history entry, and return a pointer to that entry. If there is no previous entry, return a `NULL` pointer.

`HIST_ENTRY * next_history (`*void*`)` [Function]

Move the current history offset forward to the next history entry, and return the a pointer to that entry. If there is no next entry, return a `NULL` pointer.

2.3.5 Searching the History List

These functions allow searching of the history list for entries containing a specific string. Searching may be performed both forward and backward from the current history position. The search may be *anchored*, meaning that the string must match at the beginning of the history entry.

`int history_search (`*const char *string, int direction*`)` [Function]

Search the history for *string*, starting at the current history offset. If *direction* is less than 0, then the search is through previous entries, otherwise through subsequent entries. If *string* is found, then the current history index is set to that history entry, and the value returned is the offset in the line of the entry where *string* was found. Otherwise, nothing is changed, and a -1 is returned.

`int history_search_prefix (`*const char *string, int direction*`)` [Function]

Search the history for *string*, starting at the current history offset. The search is anchored: matching lines must begin with *string*. If *direction* is less than 0, then the search is through previous entries, otherwise through subsequent entries. If *string* is found, then the current history index is set to that entry, and the return value is 0. Otherwise, nothing is changed, and a -1 is returned.

`int history_search_pos (`*const char *string, int direction, int pos*`)` [Function]

Search for *string* in the history list, starting at *pos*, an absolute index into the list. If *direction* is negative, the search proceeds backward from *pos*, otherwise forward. Returns the absolute index of the history element where *string* was found, or -1 otherwise.

2.3.6 Managing the History File

The History library can read the history from and write it to a file. This section documents the functions for managing a history file.

`int read_history (`*const char *filename*`)` [Function]

Add the contents of *filename* to the history list, a line at a time. If *filename* is `NULL`, then read from '`~/.history`'. Returns 0 if successful, or `errno` if not.

`int read_history_range (`*const char *filename, int from, int to*`)` [Function]

Read a range of lines from *filename*, adding them to the history list. Start reading at line *from* and end at *to*. If *from* is zero, start at the beginning. If *to* is less than *from*, then read until the end of the file. If *filename* is `NULL`, then read from '`~/.history`'. Returns 0 if successful, or `errno` if not.

`int write_history (`*const char *filename*`)` [Function]

Write the current history to *filename*, overwriting *filename* if necessary. If *filename* is `NULL`, then write the history list to '`~/.history`'. Returns 0 on success, or `errno` on a read or write error.

`int append_history` (*int nelements, const char *filename*) [Function]

Append the last *nelements* of the history list to *filename*. If *filename* is `NULL`, then append to '`~/.history`'. Returns 0 on success, or `errno` on a read or write error.

`int history_truncate_file` (*const char *filename, int nlines*) [Function]

Truncate the history file *filename*, leaving only the last *nlines* lines. If *filename* is `NULL`, then '`~/.history`' is truncated. Returns 0 on success, or `errno` on failure.

2.3.7 History Expansion

These functions implement history expansion.

`int history_expand` (*char *string, char **output*) [Function]

Expand *string*, placing the result into *output*, a pointer to a string (see Section 1.1 [History Interaction], page 1). Returns:

`0` If no expansions took place (or, if the only change in the text was the removal of escape characters preceding the history expansion character);

`1` if expansions did take place;

`-1` if there was an error in expansion;

`2` if the returned line should be displayed, but not executed, as with the `:p` modifier (see Section 1.1.3 [Modifiers], page 2).

If an error ocurred in expansion, then *output* contains a descriptive error message.

`char * get_history_event` (*const char *string, int *cindex, int qchar*) [Function]

Returns the text of the history event beginning at *string* `+` **cindex*. **cindex* is modified to point to after the event specifier. At function entry, *cindex* points to the index into *string* where the history event specification begins. *qchar* is a character that is allowed to end the event specification in addition to the "normal" terminating characters.

`char ** history_tokenize` (*const char *string*) [Function]

Return an array of tokens parsed out of *string*, much as the shell might. The tokens are split on the characters in the *history_word_delimiters* variable, and shell quoting conventions are obeyed.

`char * history_arg_extract` (*int first, int last, const char *string*) [Function]

Extract a string segment consisting of the *first* through *last* arguments present in *string*. Arguments are split using `history_tokenize`.

2.4 History Variables

This section describes the externally-visible variables exported by the GNU History Library.

`int history_base` [Variable]

The logical offset of the first entry in the history list.

`int history_length` [Variable]

The number of entries currently stored in the history list.

`int history_max_entries` [Variable]

The maximum number of history entries. This must be changed using `stifle_history()`.

`int history_write_timestamps` [Variable]

If non-zero, timestamps are written to the history file, so they can be preserved between sessions. The default value is 0, meaning that timestamps are not saved.

The current timestamp format uses the value of *history_comment_char* to delimit timestamp entries in the history file. If that variable does not have a value (the default), timestamps will not be written.

`char history_expansion_char` [Variable]

The character that introduces a history event. The default is '`!`'. Setting this to 0 inhibits history expansion.

`char history_subst_char` [Variable]

The character that invokes word substitution if found at the start of a line. The default is '`^`'.

`char history_comment_char` [Variable]

During tokenization, if this character is seen as the first character of a word, then it and all subsequent characters up to a newline are ignored, suppressing history expansion for the remainder of the line. This is disabled by default.

`char * history_word_delimiters` [Variable]

The characters that separate tokens for `history_tokenize()`. The default value is `" \t\n()<>;&|"`.

`char * history_search_delimiter_chars` [Variable]

The list of additional characters which can delimit a history search string, in addition to space, TAB, '`:`' and '`?`' in the case of a substring search. The default is empty.

`char * history_no_expand_chars` [Variable]

The list of characters which inhibit history expansion if found immediately following *history_expansion_char*. The default is space, tab, newline, carriage return, and '`=`'.

`int history_quotes_inhibit_expansion` [Variable]

If non-zero, single-quoted words are not scanned for the history expansion character. The default value is 0.

`rl_linebuf_func_t * history_inhibit_expansion_function` [Variable]

This should be set to the address of a function that takes two arguments: a `char *` (*string*) and an `int` index into that string (*i*). It should return a non-zero value if the history expansion starting at *string[i]* should not be performed; zero if the expansion should be done. It is intended for use by applications like Bash that use the history expansion character for additional purposes. By default, this variable is set to `NULL`.

2.5 History Programming Example

The following program demonstrates simple use of the GNU History Library.

```
#include <stdio.h>
#include <readline/history.h>

main (argc, argv)
     int argc;
     char **argv;
{
  char line[1024], *t;
  int len, done = 0;

  line[0] = 0;

  using_history ();
  while (!done)
    {
      printf ("history$ ");
      fflush (stdout);
      t = fgets (line, sizeof (line) - 1, stdin);
      if (t && *t)
        {
          len = strlen (t);
          if (t[len - 1] == '\n')
            t[len - 1] = '\0';
        }

      if (!t)
        strcpy (line, "quit");

      if (line[0])
        {
          char *expansion;
          int result;

          result = history_expand (line, &expansion);
          if (result)
            fprintf (stderr, "%s\n", expansion);

          if (result < 0 || result == 2)
            {
              free (expansion);
              continue;
            }

          add_history (expansion);
          strncpy (line, expansion, sizeof (line) - 1);
          free (expansion);
        }

      if (strcmp (line, "quit") == 0)
        done = 1;
      else if (strcmp (line, "save") == 0)
        write_history ("history_file");
      else if (strcmp (line, "read") == 0)
        read_history ("history_file");
      else if (strcmp (line, "list") == 0)
```

```
          {
            register HIST_ENTRY **the_list;
            register int i;

            the_list = history_list ();
            if (the_list)
              for (i = 0; the_list[i]; i++)
                printf ("%d: %s\n", i + history_base, the_list[i]->line);
          }
        else if (strncmp (line, "delete", 6) == 0)
          {
            int which;
            if ((sscanf (line + 6, "%d", &which)) == 1)
              {
                HIST_ENTRY *entry = remove_history (which);
                if (!entry)
                  fprintf (stderr, "No such entry %d\n", which);
                else
                  {
                    free (entry->line);
                    free (entry);
                  }
              }
            else
              {
                fprintf (stderr, "non-numeric arg given to ‘delete’\n");
              }
          }
      }
  }
```

Appendix A GNU Free Documentation License

0. 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 `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.

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 (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.

Appendix B Concept Index

A

anchored search 7

E

event designators 1

H

history events 1
history expansion 1
History Searching 7

Appendix C Function and Variable Index

A

add_history ... 5
add_history_time ... 5
append_history ... 8

C

clear_history ... 6
current_history ... 6

F

free_history_entry ... 5

G

get_history_event ... 8

H

history_arg_extract ... 8
history_base ... 8
history_comment_char ... 9
history_expand ... 8
history_expansion_char ... 9
history_get ... 6
history_get_history_state ... 5
history_get_time ... 6
history_inhibit_expansion_function ... 9
history_is_stifled ... 6
history_length ... 8
history_list ... 6
history_max_entries ... 9
history_no_expand_chars ... 9
history_quotes_inhibit_expansion ... 9
history_search ... 7
history_search_delimiter_chars ... 9
history_search_pos ... 7
history_search_prefix ... 7
history_set_history_state ... 5
history_set_pos ... 6
history_subst_char ... 9
history_tokenize ... 8
history_total_bytes ... 6
history_truncate_file ... 8
history_word_delimiters ... 9
history_write_timestamps ... 9

N

next_history ... 7

P

previous_history ... 6

R

read_history ... 7
read_history_range ... 7
remove_history ... 5
replace_history_entry ... 5

S

stifle_history ... 6

U

unstifle_history ... 6
using_history ... 5

W

where_history ... 6
write_history ... 7