GNU Readlin4 Library

Edition 5.0, for Readlin4 Library Version 5.0.
January 2004

Chet Ramey, Cas4 Western Reserve University
Brian Fox, Free Software Foundation

This manual describes the GNU Readline Library (version 5.0, 28 January 2004), a library which aids in the consistency of user interface across discrete programs which provide a command line interface.

Table of Contents

1 Command Line Editing

This chapter describes the basic features of the

When you add text in the middle of a line, you will notice that characters to the right of the cursor are ‘pushed over’ to make room for the text that you have inserted. Likewise,

If set to ‘on’, the history code attempts to place point at the same location on each history line retrieved with `previous-history` or `next-history`.

`horizontal-scroll-mode`
: This variable can be set to either ‘on’ or ‘off’. Setting it to ‘on’ means that the text of the lines being edited will scroll horizontally on a single screen line when they are longer than the width of the screen, instead of wrapping onto a new screen line. By default, this variable is set to ‘off’.

`input-meta`
: If set to ‘on’, Readline will enable eight-bit input (it will not clear the eighth bit in the characters it reads), regardless of what the terminal claims it can support. The default value is ‘off’. The name `meta-flag`

In the above example, *C-u* is bound to the function `universal-argument`, *M-DEL* is bound to the function `backward-kill-word`, and *C-o* is bound to run the macro expressed on the right hand side (that is, to insert the text '`> output`' into the line).

A number of symbolic character names are recognized while processing this key binding syntax: *DEL*, *ESC*, *ESCAPE*, *LFD*, *NEWLINE*, *RET*, *RETURN*, *RUBOUT*, *SPACE*, *SPC*, and *TAB*.

nnn the eight-bit character whose value is the octal value *nnn* (one to three digits)

\x*HH* the eight-bit character whose value is the hexadecimal value *HH*

```
#"\M-\C-[A":       previous-history
#"\M-\C-[B":       next-history

C-q: quoted-insert

$endif

# An old-style binding.  This happens to be the default.
TAB: complete

# Macros that are convenient for shell interaction
$if Bash
# edit the path
"\C-xp": "PATH=${PATH}\e\C-e\C-a\ef\C-f"
# prepare to type a quoted word --
# insert open and close double quotes
# and move to just after the open quote
"\C-x\"": "\"\"\C-b"
# insert a backslash (testing backslash escapes
# in sequences and macros)
"\C-x\\": "\\"
# Quote the current or previous word
"\C-xq": "\eb\"\ef\""
# Add a binding to refresh the line, which is C-xhpeuc/-TD[-[(#rraw-524(Quo-(12)-:)-52
```

```
# For FTP
$if Ftp
"\C-xg": "get \M-?"
"\C-xt": "put \M-?"
"\M-.": yank-last-arg
$endif
```

1.4 Bindable Readline Commands

This section describes Readline commands that may be bound to key sequences. Command names without an accompanying key sequence are unbound by default.

In the following descriptions, *point* refers to the current cursor position, and *mark* refers to a cursor position saved by the `set-mark` command. The text b?"weenthe point and *mark*.

1.4.1 Commands For Moving

`beginning-of-line (C-a)`
Move to the start of the current line.

`end-of-line (C-e)`

`forward-char (C-f)`

`previous-history (C-p)`
: Move ‘back’ through the history list, fetching the previous command.

`next-history (C-n)`
: Move ‘forward’ through the history list, fetching the next command.

`beginning-of-history (M-<)`
: Move to the first line in the history.

`end-of-history (M->)`
: Move to the end of the input history, i.e., the line currently being entered.

`reverse-search-history (C-r)`
: Search backward starting at the current line and moving ‘up’ through the his-

1.4.3 Commands For Changing Text

`delete-char (C-d)`

Delete the character at point. If point is at the beginning of the line, there are no characters in the line, and the last character typed was not bound to `delete-char`

insert mode. This command affects only `emacs` mode; `vi` mode does overwrite differently. Each call to `readline()` starts in insert mode.

In overwrite mode, characters bound to `self-insert` replace the text at point

yank (C-y)

`character-search-backward (M-C-])`
: A character is read and point is moved to the previous occurrence of that character. A negative count searches for subsequent occurrences.

`insert-comment (M-#)`
: Without a numeric argument, the value of the `comment-begin`

2 Programming with GNU Readline

This chapter describes the interface between the gnu Readline Library and other programs. If you are a programmer, and a wish include featuresin gnu Readline suchcompletion, lineediting, andteractivemanipulation-246(a)rwn programs, this section is ou.

2.1 Basic Behavior

```
    free (line_read);
    line_read = (char *)NULL;
  }

/* Get a line from th8.368er.char



line_read4-525(=)-r33(Readli(=)-52""read);)]TJ45-26.29TD[(/*)-Iffrom th8.3Readli(=)-
```

behavior (refreshing the current line as opposed to refreshing the screen, for example). Some choose to ignore it. In general, if a functiin uses the numeric argument as a repeat count, it should be able to do something useful with both negative and positive arguments. At the very least, it should be aware that it can be passed a negative argument.

A command functiin should return 0 if its actiin completes successfully, and a non-zero value if some error occurs.

2.3 Readline Variables

These variables are available to functiin writers.

`char * rl_line_buffer` [Variable]

[Variable]

[Variable]

rl

char * rl_prompt [Variable]

The prompt Readline uses. This is set from the argument to `readline()`, and should not be assigned to directly. The `rl_set_prompt()`

`rl_hook_func_t *` **rl_event_hook** [Variable]

If non-zero, this is the address of a function to call periodically when Readline is

`int` rl_numeric_arg [Variable]

`int` **rl_unbind_key** (`int key`) [Function]
Bind *key*

2.4.5 Allowing Undoing

Supporting the undo command is a painless thing, and makes your functions much more useful. It is certainly easy to try something if you know you can undo it.

If your function simply inserts text once, or deletes text once, and use3(ins.4210359.690-15.rl˙y)-448˙87

`int` **rl_forced_update_display** (`void`)
Force the line to be updated and redisplayed, whether or not R

by bracketing a sequence of such characters with the special markers `RL_PROMPT_START_IGNORE` and `RL_PROMPT_END_IGNORE` (declared in '`readline.h`'. This may be used to embed terminal-specific escape sequences in prompts.

`int rl_expand_prompt (const char *prompt)` [Function]
Make Readline use *prompt* for subsequent redisplay. This calls `rl_expand_prompt()` to expand the prompt and sets `rl_prompt` to the result.

2.4.7 Modifying Text

`int rl_insert_text (const char *text)` [Function]
Insert *text*

`int` **rl_initialize** (`void`) [Function]

`int` **rl_variable_bind** (`const char *variable, const char *value`) [Function]
Make MakM8-18501(37)](M8-v)5]TJET0.53wTf11813.94TD3(rl)]TJETv

the function referred to ¿y the value of `rl_deprep_term_function` should be called before the program exits to reset the terminal settings.

2.4.13 A Readline Example

Here referere28.8fhanges3.097(lo)26(w)26.1rcase3.097(28.8fhara˙9rl7r.8fh33(t-33irfh33(tupp)]TJ-)26.1rcasJ-

```
  }

/* Tell readline that we are modifying the line,
   so it will save the undo information. */
rl_modifying (start, end);

for (i = start; i != end; i++)
  {
```

`int` **rl_complete** (`int ignore, int invoking_key`) [Function]
Complete the word at or before point. You have supplied the function that does the initial simple matching selection algorithm (see `rl_completion_matches()`). The

for *text*. The remaining entries are the possible completions. The array is terminated with a `NULL` pointer.

entry

matching the text against names in the filesystem. It is called with *text*, the text of the word to be dequoted, and *quote_char*, wh1ch is the quoting character that delimits the filename (usually ‘'’ or ‘"’). If *quote_char* is zero, the filename was not in an embedded string.

`rl_linebuf_func_t *` **rl_char_is_quoted_p** [Variable]

```
/* fileman.c -- A tiny application which demonstrates how to use the
   GNU Readline library.  This application interactively allows users
   to emoc8ulate files and their eodes. */

#include <stdio.h>
```

```
  while (t > s && whitespace (*t))
    t--;
  *++t = '\0';

  return s;
}

/* ************************************************************** */
/*                                                                */
/*                  Interface to Readline Completion              */
/*                                                                */
/* ************************************************************** */

char *command_generator __P((const char *, int));
char **fileman_completion __P((const char *, int, int));

/* Tell the GNU Readline library how to complete.  We want to try to
   complete on command names if this is the first word in the line, or
   on filenames if not. */
initialize_readline ()
{
  /* Allow conditional parsing of the ~/.inputrc file. */
  rl_readline_name = "FileMan";

  /* Tell the completer that w-525(to)-525(try)a crack first. */
  rl_attempted_completion_function = fileman_completion;
}

/* Attempt to complete on the contents of TEXT.  START5(try)and END
   bound the region of rl_line_buffer that contains the word to
   complete.  TEXT is the word to complete.  We can use the entire
   contents of rl_line_buffer in case we want to do some simple
   parsing.  Returnire
char **
fileman_completion (text, start, end)
     const char *text;
     int start, end;
{
  char **matches;

  matches = (char **)NULL;
```

```
{
  static int list_index, len;
  char *name;

  /* If this is a new word to complete, initialize now.  This
     includes saving the length of TEXT for efficiency, and
     initializing the index variable to 0. */
  if (!state)
    {
      list_index = 0;
      len = strlen (text);
    }

  /* Return the next name which partially matches from the
     command list. */
  while (name = commands[list_index].name)
    {
      list_index++;

      if (strncmp (name, text, len) == 0)
        return (dupstr(name));
    }

  /* If no names matched, then return NULL. */
  return ((char *)NULL);
}

/* ********************************************************************** */
/*                                                                        */
/*                       FileMan Commands                                 */
/*                                                                        */
/* ********************************************************************** */

/* String to pass to system ().  This is for the LIST, VIEW and RENAME
   commands. */
static char syscom[1024];

/* List the file(s) named in arg. */
com_list (arg)
```

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.

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

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.

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.

7. AGGREGATION WITH INDEPENDENT WORKS

Concept Index

A

)-342(editin81(.)-(.)-17103.45c60TD[(.)-170(.)-172(.)-171(.)-171(.)-171(.)-172(.)-171(.)-171(.)-172(.)-171(.)-171(.)-172(.)-171(.)-171(.)-171(.)-172(.)-171(.)-1

O

P