GNU Readline

Introduction

The readline functions implement an interface to the GNU Readline library. These are functions that provide editable command lines. An example being the way Bash allows you to use the arrow keys to insert characters or scroll through command history. Because of the interactive nature of this library, it will be of little use for writing Web applications, but may be useful when writing scripts used from a command line.

Note
This extension is not available on Windows platforms.

Installing/Configuring

Requirements

To use the readline functions, you need to install libreadline. You can find libreadline on the home page of the GNU Readline project, at » http://cnswww.cns.cwru.edu/~chet/readline/rltop.html. It's maintained by Chet Ramey, who's also the author of Bash.

You can also use these functions with the libedit library, a non-GPL replacement for the readline library. The libedit library is BSD licensed and available for download from » http://www.thrysoee.dk/editline/.

Installation

To use these functions you must compile the CGI or CLI version of PHP with readline support. You need to configure PHP *--with-readline[=DIR]*. In order you want to use the libedit readline replacement, configure PHP *--with-libedit[=DIR]*.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Readline Functions

readline_add_history

readline_add_history -- Adds a line to the history

Description

bool **readline_add_history** (string `$line`)

This function adds a line to the command line history.

Parameters

`line`
: The line to be added in the history.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

readline_callback_handler_install

readline_callback_handler_install -- Initializes the readline callback interface and terminal, prints the prompt and returns immediately

Description

bool **readline_callback_handler_install** (string `$prompt`, callback `$callback`)

Sets up a readline callback interface then prints *`prompt`* and immediately returns. Calling this function twice without removing the previous callback interface will automatically and conveniently overwrite the old interface.

The callback feature is useful when combined with stream_select() as it allows interleaving of IO and user input, unlike readline().

Parameters

`prompt`
The prompt message.

`callback`
The *`callback`* function takes one parameter; the user input returned.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #1 - Readline Callback Interface Example

```
<?php
function rl_callback($ret)
{
   global $c, $prompting;

   echo "You entered: $ret\n";
   $c++;

   if ($c > 10) {
       $prompting = false;
       readline_callback_handler_remove();
   } else {
       readline_callback_handler_install("[$c] Enter something: ",
'rl_callback');
   }
```

```
}

$c = 1;
$prompting = true;

readline_callback_handler_install("[$c] Enter something: ", 'rl_callback');

while ($prompting) {
   $w = NULL;
   $e = NULL;
   $n = stream_select($r = array(STDIN), $w, $e, null);
   if ($n && in_array(STDIN, $r)) {
       // read a character, will call the callback when a newline is entered
       readline_callback_read_char();
   }
}

echo "Prompting disabled. All done.\n";
?>
```

See Also

- readline_callback_handler_remove()
- readline_callback_read_char()
- stream_select()

readline_callback_handler_remove

readline_callback_handler_remove -- Removes a previously installed callback handler and restores terminal settings

Description

bool **readline_callback_handler_remove** (void)

Removes a previously installed callback handler and restores terminal settings.

Return Values

Returns **TRUE** if a previously installed callback handler was removed, or **FALSE** if one could not be found.

Examples

See readline_callback_handler_install() for an example of how to use the readline callback interface.

See Also

- readline_callback_handler_install()
- readline_callback_read_char()

readline_callback_read_char

readline_callback_read_char -- Reads a character and informs the readline callback interface when a line is received

Description

void **readline_callback_read_char** (void)

Reads a character of user input. When a line is received, this function informs the readline callback interface installed using readline_callback_handler_install() that a line is ready for input.

Return Values

No value is returned.

Examples

See readline_callback_handler_install() for an example of how to use the readline callback interface.

See Also

- readline_callback_handler_install()
- readline_callback_handler_remove()

readline_clear_history

readline_clear_history -- Clears the history

Description

bool **readline_clear_history** (void)

This function clears the entire command line history.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

readline_completion_function

readline_completion_function -- Registers a completion function

Description

bool **readline_completion_function** (callback `$function`)

This function registers a completion function. This is the same kind of functionality you'd get if you hit your tab key while using Bash.

Parameters

`function`
: You must supply the name of an existing function which accepts a partial command line and returns an array of possible matches.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

readline_info

readline_info -- Gets/sets various internal readline variables

Description

mixed **readline_info** ([string `$varname` [, string `$newvalue`]])

Gets or sets various internal readline variables.

Parameters

`varname`
: A variable name.

`newvalue`
: If provided, this will be the new value of the setting.

Return Values

If called with no parameters, this function returns an array of values for all the setting readline uses. The elements will be indexed by the following values: done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, and terminal_name.

If called with one or two parameters, the old value is returned.

readline_list_history

readline_list_history -- Lists the history

Description

array **readline_list_history** (void)

Gets the entire command line history.

Return Values

Returns an array of the entire command line history. The elements are indexed by integers starting at zero.

readline_on_new_line

readline_on_new_line -- Inform readline that the cursor has moved to a new line

Description

void **readline_on_new_line** (void)

Tells readline that the cursor has moved to a new line.

Return Values

No value is returned.

readline_read_history

readline_read_history -- Reads the history

Description

bool **readline_read_history** ([string `$filename`])

This function reads a command history from a file.

Parameters

`filename`
: Path to the filename containing the command history.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

readline_redisplay

readline_redisplay -- Redraws the display

Description

void **readline_redisplay** (void)

Redraws readline to redraw the display.

Return Values

No value is returned.

readline_write_history

readline_write_history -- Writes the history

Description

bool **readline_write_history** ([string `$filename`])

This function writes the command history to a file.

Parameters

`filename`
: Path to the saved file.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

readline

readline -- Reads a line

Description

string **readline** ([string `$prompt`])

Reads a single line from the user. You must add this line to the history yourself using readline_add_history().

Parameters

`prompt`
You may specify a string with which to prompt the user.

Return Values

Returns a single string from the user. The line returned has the ending newline removed.

Examples

Example #2 - readline() Example

```
<?php
//get 3 commands from user
for ($i=0; $i < 3; $i++) {
       $line = readline("Command: ");
       readline_add_history($line);
}

//dump history
print_r(readline_list_history());

//dump variables
print_r(readline_info());
?>
```