In Files

curses/curses.c (./curses/curses_c.html?TB_iframe=true&height=550&width=785)

Namespace

MODULE Curses::Key (Curses/Key.html)
CLASS Curses::MouseEvent (Curses/MouseEve
CLASS Curses::Window (Curses/Window.html)

Methods

::ESCDELAY (#method-c-ESCDELAY)
::ESCDELAY= (#method-c-ESCDELAY-3D)
::TABSIZE (#method-c-TABSIZE)
::TABSIZE= (#method-c-TABSIZE-3D)
::addch (#method-c-addch)
::addstr (#method-c-addstr)
::attroff (#method-c-attroff)
::attron (#method-c-attron)
::attrset (#method-c-attrset)
::beep (#method-c-beep)
::bkgd (#method-c-bkgd)
::bkgdset (#method-c-bkgdset)
::can_change_color? (#method-c-can_change_
::cbreak (#method-c-cbreak)
::clear (#method-c-clear)
::close_screen (#method-c-close_screen)
::closed? (#method-c-closed-3F)
::clrtoeol (#method-c-clrtoeol)
::color_content (#method-c-color_content)
::color_pair (#method-c-color_pair)
::color_pairs (#method-c-color_pairs)
::colors (#method-c-colors)
::cols (#method-c-cols)
::crmode (#method-c-crmode)
::curs_set (#method-c-curs_set)

Curses

Description

An implementation of the CRT screen handling and optimization library.

Structures and such

Classes

- Curses::Window (Curses/Window.html) - class with the means to draw a window or box
- Curses::MouseEvent (Curses/MouseEvent.html) - class for collecting mouse events

Modules

Curses (Curses.html) :

The curses implementation

Curses::Key (Curses/Key.html) :

Collection of constants for keypress events

Examples

- hello.rb

```
#!/usr/local/bin/ruby


require "curses"
include Curses
```

::def_prog_mode (#method-c-def_prog_mode)
::delch (#method-c-delch)
::deleteln (#method-c-deleteln)
::doupdate (#method-c-doupdate)
::echo (#method-c-echo)
::flash (#method-c-flash)
::getch (#method-c-getch)
::getmouse (#method-c-getmouse)
::getstr (#method-c-getstr)
::has_colors? (#method-c-has_colors-3F)
::inch (#method-c-inch)
::init_color (#method-c-init_color)
::init_pair (#method-c-init_pair)
::init_screen (#method-c-init_screen)
::insch (#method-c-insch)
::insertln (#method-c-insertln)
::keyname (#method-c-keyname)
::lines (#method-c-lines)
::mouseinterval (#method-c-mouseinterval)
::mousemask (#method-c-mousemask)
::nl (#method-c-nl)
::nocbreak (#method-c-nocbreak)
::nocrmode (#method-c-nocrmode)
::noecho (#method-c-noecho)
::nonl (#method-c-nonl)
::noraw (#method-c-noraw)
::pair_content (#method-c-pair_content)
::pair_number (#method-c-pair_number)
::raw (#method-c-raw)
::refresh (#method-c-refresh)
::reset_prog_mode (#method-c-reset_prog_mo
::resize (#method-c-resize)
::resizeterm (#method-c-resizeterm)
::scrl (#method-c-scrl)
::setpos (#method-c-setpos)
::setscrreg (#method-c-setscrreg)
::standend (#method-c-standend)
::standout (#method-c-standout)
::start_color (#method-c-start_color)
::stdscr (#method-c-stdscr)
::timeout= (#method-c-timeout-3D)
::ungetch (#method-c-ungetch)
::ungetmouse (#method-c-ungetmouse)
::use_default_colors (#method-c-use_default_c

Class/Module Index

Curses (./Curses.html)

```
def show_message(message)
  width = message.length + 6
  win = Window.new(5, width,
              (lines - 5) / 2, (cols - width) /
  win.box(||, --)
  win.setpos(2, 3)
  win.addstr(message)
  win.refresh
  win.getch
  win.close
end

init_screen
begin
  crmode
#  show_message("Hit any key")
  setpos((lines - 5) / 2, (cols - 10) / 2)
  addstr("Hit any key")
  refresh
  getch
  show_message("Hello, World!")
  refresh
ensure
  close_screen
end
```

- rain.rb

```
#!/usr/local/bin/ruby
# rain for a curses test

require "curses"
include Curses

def onsig(sig)
  close_screen
  exit sig
end
```

Curses::Key (./Curses/Key.html)
Curses::MouseEvent (./Curses/MouseEvent.htn
Curses::Window (./Curses/Window.html)
FileViewer (./FileViewer.html)
Object (./Object.html)

```
def ranf
  rand(32767).to_f / 32767
end

# main #
for i in 1 .. 15  # SIGHUP .. SIGTERM
  if trap(i, "SIG_IGN") != 0 then  # 0 for SIG_I
    trap(i) {|sig| onsig(sig) }
  end
end

init_screen
nl
noecho
srand

xpos = {}
ypos = {}
r = lines - 4
c = cols - 4
for i in 0 .. 4
  xpos[i] = (c * ranf).to_i + 2
  ypos[i] = (r * ranf).to_i + 2
end

i = 0
while TRUE
  x = (c * ranf).to_i + 2
  y = (r * ranf).to_i + 2

  setpos(y, x); addstr(".")

  setpos(ypos[i], xpos[i]); addstr("o")

  i = if i == 0 then 4 else i - 1 end
  setpos(ypos[i], xpos[i]); addstr("O")
```

```
    i = if i == 0 then 4 else i - 1 end
    setpos(ypos[i] - 1, xpos[i]);      addstr("-")
    setpos(ypos[i],     xpos[i] - 1); addstr("|.|"
    setpos(ypos[i] + 1, xpos[i]);      addstr("-")

    i = if i == 0 then 4 else i - 1 end
    setpos(ypos[i] - 2, xpos[i]);       addstr("-"
    setpos(ypos[i] - 1, xpos[i] - 1);  addstr("/ \
    setpos(ypos[i],     xpos[i] - 2); addstr("| O
    setpos(ypos[i] + 1, xpos[i] - 1); addstr("\\ /
    setpos(ypos[i] + 2, xpos[i]);       addstr("-"

    i = if i == 0 then 4 else i - 1 end
    setpos(ypos[i] - 2, xpos[i]);       addstr(" "
    setpos(ypos[i] - 1, xpos[i] - 1);  addstr("
    setpos(ypos[i],     xpos[i] - 2); addstr("
    setpos(ypos[i] + 1, xpos[i] - 1);  addstr("
    setpos(ypos[i] + 2, xpos[i]);       addstr(" "

    xpos[i] = x
    ypos[i] = y
    refresh
    sleep(0.5)
  end

  # end of main
```

Constants

ALL_MOUSE_EVENTS:
report all button state changes

See ::getmouse (Curses.html#method-c-getmouse)

A_ALTCHARSET:
Alternate character set

See ::attrset (Curses.html#method-c-attrset)

A_ATTRIBUTES:
Curses::Window#inch (Curses/Window.html#method-i-inch)

A_BLINK:
Blinking

See ::attrset (Curses.html#method-c-attrset)

A_BOLD:
Extra bright or bold

See ::attrset (Curses.html#method-c-attrset)

A_CHARTEXT:
Bit-mask to extract a character

See ::attrset (Curses.html#method-c-attrset)

A_COLOR:
Curses::Window#inch (Curses/Window.html#method-i-inch)

A_DIM:
Half bright

See ::attrset (Curses.html#method-c-attrset)

A_HORIZONTAL:
horizontal highlight

Check system curs_attr(3x) for support

A_INVIS:
Invisible or blank mode

See ::attrset (Curses.html#method-c-attrset)

A_LEFT:
left highlight

Check system curs_attr(3x) for support

A_LOW:
low highlight

Check system curs_attr(3x) for support

A_NORMAL:
Normal display (no highlight)

See ::attrset (Curses.html#method-c-attrset)

`A_PROTECT`:
Protected mode

See ::attrset (Curses.html#method-c-attrset)

`A_REVERSE`:
Reverse video

See ::attrset (Curses.html#method-c-attrset)

`A_RIGHT`:
right highlight

Check system curs_attr(3x) for support

`A_STANDOUT`:
Best highlighting mode of the terminal.

See ::attrset (Curses.html#method-c-attrset)

`A_TOP`:
top highlight

Check system curs_attr(3x) for support

`A_UNDERLINE`:
Underlining

See ::attrset (Curses.html#method-c-attrset)

`A_VERTICAL`:
vertical highlight

Check system curs_attr(3x) for support

`BUTTON1_CLICKED`:
mouse button 1 clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON1_DOUBLE_CLICKED`:
mouse button 1 double clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON1_PRESSED`:

mouse button 1 down

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON1_RELEASED`:
mouse button 1 up

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON1_TRIPLE_CLICKED`:
mouse button 1 triple clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON2_CLICKED`:
mouse button 2 clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON2_DOUBLE_CLICKED`:
mouse button 2 double clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON2_PRESSED`:
mouse button 2 down

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON2_RELEASED`:
mouse button 2 up

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON2_TRIPLE_CLICKED`:
mouse button 2 triple clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON3_CLICKED`:
mouse button 3 clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON3_DOUBLE_CLICKED`:
mouse button 3 double clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON3_PRESSED`:
mouse button 3 down

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON3_RELEASED`:
mouse button 3 up

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON3_TRIPLE_CLICKED`:
mouse button 3 triple clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON4_CLICKED`:
mouse button 4 clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON4_DOUBLE_CLICKED`:
mouse button 4 double clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON4_PRESSED`:
mouse button 4 down

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON4_RELEASED`:
mouse button 4 up

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON4_TRIPLE_CLICKED`:
mouse button 4 triple clicked

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON_ALT`:
alt was down during button state change

See ::getmouse (Curses.html#method-c-getmouse)

`BUTTON_CTRL`:
control was down during button state change

See ::getmouse (Curses.html#method-c-getmouse)

BUTTON_SHIFT:
shift was down during button state change

See ::getmouse (Curses.html#method-c-getmouse)

COLORS:
Number of the colors available

COLOR_BLACK:
Value of the color black

COLOR_BLUE:
Value of the color blue

COLOR_CYAN:
Value of the color cyan

COLOR_GREEN:
Value of the color green

COLOR_MAGENTA:
Value of the color magenta

COLOR_RED:
Value of the color red

COLOR_WHITE:
Value of the color white

COLOR_YELLOW:
Value of the color yellow

KEY_A1:
Upper left of keypad

KEY_A3:
Upper right of keypad

KEY_B2:
Center of keypad

KEY_BACKSPACE:
Backspace

KEY_BEG:

Beginning key

`KEY_BREAK`:
Break key

`KEY_BTAB`:
Back tab key

`KEY_C1`:
Lower left of keypad

`KEY_C3`:
Lower right of keypad

`KEY_CANCEL`:
Cancel key

`KEY_CATAB`:
Clear all tabs

`KEY_CLEAR`:
Clear Screen

`KEY_CLOSE`:
Close key

`KEY_COMMAND`:
Cmd (command) key

`KEY_COPY`:
Copy key

`KEY_CREATE`:
Create key

`KEY_CTAB`:
Clear tab

`KEY_DC`:
Delete character

`KEY_DL`:
Delete line

`KEY_DOWN`:
the down arrow key

KEY_EIC:
Enter insert char mode

KEY_END:
End key

KEY_ENTER:
Enter or send

KEY_EOL:
Clear to end of line

KEY_EOS:
Clear to end of screen

KEY_EXIT:
Exit key

KEY_FIND:
Find key

KEY_HELP:
Help key

KEY_HOME:
Home key (upward+left arrow)

KEY_IC:
Insert char or enter insert mode

KEY_IL:
Insert line

KEY_LEFT:
the left arrow key

KEY_LL:
Home down or bottom (lower left)

KEY_MARK:
Mark key

KEY_MAX:
The maximum allowed curses key value.

KEY_MESSAGE:
Message key

KEY_MIN:

The minimum allowed curses key value.

KEY_MOUSE:

Mouse event read

KEY_MOVE:

Move key

KEY_NEXT:

Next object key

KEY_NPAGE:

Next page

KEY_OPEN:

Open key

KEY_OPTIONS:

Options key

KEY_PPAGE:

Previous page

KEY_PREVIOUS:

Previous object key

KEY_PRINT:

Print or copy

KEY_REDO:

Redo key

KEY_REFERENCE:

Reference key

KEY_REFRESH:

Refresh key

KEY_REPLACE:

Replace key

KEY_RESET:

Reset or hard reset

KEY_RESIZE:

Screen Resized

`KEY_RESTART`:
Restart key

`KEY_RESUME`:
Resume key

`KEY_RIGHT`:
the right arrow key

`KEY_SAVE`:
Save key

`KEY_SBEG`:
Shifted beginning key

`KEY_SCANCEL`:
Shifted cancel key

`KEY_SCOMMAND`:
Shifted command key

`KEY_SCOPY`:
Shifted copy key

`KEY_SCREATE`:
Shifted create key

`KEY_SDC`:
Shifted delete char key

`KEY_SDL`:
Shifted delete line key

`KEY_SELECT`:
Select key

`KEY_SEND`:
Shifted end key

`KEY_SEOL`:
Shifted clear line key

`KEY_SEXIT`:
Shifted exit key

`KEY_SF`:
Scroll 1 line forward

`KEY_SFIND`:
Shifted find key

`KEY_SHELP`:
Shifted help key

`KEY_SHOME`:
Shifted home key

`KEY_SIC`:
Shifted input key

`KEY_SLEFT`:
Shifted left arrow key

`KEY_SMESSAGE`:
Shifted message key

`KEY_SMOVE`:
Shifted move key

`KEY_SNEXT`:
Shifted next key

`KEY_SOPTIONS`:
Shifted options key

`KEY_SPREVIOUS`:
Shifted previous key

`KEY_SPRINT`:
Shifted print key

`KEY_SR`:
Scroll 1 line backware (reverse)

`KEY_SREDO`:
Shifted redo key

`KEY_SREPLACE`:
Shifted replace key

`KEY_SRESET`:
Soft (partial) reset

`KEY_SRIGHT`:
Shifted right arrow key

KEY_SRSUME:
Shifted resume key

KEY_SSAVE:
Shifted save key

KEY_SSUSPEND:
Shifted suspend key

KEY_STAB:
Set tab

KEY_SUNDO:
Shifted undo key

KEY_SUSPEND:
Suspend key

KEY_UNDO:
Undo key

KEY_UP:
the up arrow key

REPORT_MOUSE_POSITION:
report mouse movement

See ::getmouse (Curses.html#method-c-getmouse)

Public Class Methods

ESCDELAY()

Returns the total time, in milliseconds, for which curses will await a character sequence, e.g., a function key

ESCDELAY=(value)

Sets the ::ESCDELAY (Curses.html#method-c-ESCDELAY) to Integer `value`

TABSIZE()

Returns the number of positions in a tab.

TABSIZE=(value)

Sets the ::TABSIZE (Curses.html#method-c-TABSIZE) to Integer `value`

addch(ch)

Add a character `ch`, with attributes, then advance the cursor.

see also the system manual for curs_addch(3)

addstr(str)

add a string of characters `str`, to the window and advance cursor

attroff(attrs)

Turns on the named attributes `attrs` without affecting any others.

See also Curses::Window#attrset (Curses/Window.html#method-i-attrset) for additional information.

attron(attrs)

Turns off the named attributes `attrs` without turning any other attributes on or off.

See also Curses::Window#attrset (Curses/Window.html#method-i-attrset) for additional information.

attrset(attrs)

Sets the current attributes of the given window to `attrs`.

see also Curses::Window#attrset (Curses/Window.html#method-i-attrset)

beep()

Sounds an audible alarm on the terminal, if possible; otherwise it flashes the screen (visual bell).

see also ::flash (Curses.html#method-c-flash)

bkgd(ch)

Window (Curses/Window.html) background manipulation routines.

Set the background property of the current and then apply the character Integer `ch` setting to every character position in that window.

see also the system manual for curs_bkgd(3)

bkgdset(ch)

Manipulate the background of the named window with character Integer `ch`

The background becomes a property of the character and moves with the character through any scrolling and insert/delete line/character operations.

see also the system manual for curs_bkgd(3)

can_change_color?()

Returns `true` or `false` depending on whether the terminal can change color attributes

cbreak()

Put the terminal into cbreak mode.

Normally, the tty driver buffers typed characters until a newline or carriage return is typed. The ::cbreak (Curses.html#method-c-cbreak) routine disables line buffering and erase/kill character-processing (interrupt and flow control characters are unaffected), making characters typed by the user immediately available to the program.

The ::nocbreak (Curses.html#method-c-nocbreak) routine returns the terminal to normal (cooked) mode.

Initially the terminal may or may not be in cbreak mode, as the mode is inherited; therefore, a program should call ::cbreak (Curses.html#method-c-cbreak) or ::nocbreak (Curses.html#method-c-nocbreak) explicitly. Most interactive programs using curses set the cbreak mode. Note that ::cbreak (Curses.html#method-c-cbreak) overrides ::raw (Curses.html#method-c-raw).

see also ::raw (Curses.html#method-c-raw)

clear()

Clears every position on the screen completely, so that a subsequent call by ::refresh (Curses.html#method-c-refresh) for the screen/window will be repainted from scratch.

close_screen()

A program should always call ::close_screen (Curses.html#method-c-close_screen) before exiting or escaping from curses mode temporarily. This routine restores tty modes, moves the cursor to the lower left-hand corner of the screen and resets the terminal into the proper non-visual mode.

Calling ::refresh (Curses.html#method-c-refresh) or ::doupdate (Curses.html#method-c-doupdate) after a temporary escape causes the program to resume visual mode.

closed?()

Returns `true` if the window/screen has been closed, without any subsequent ::refresh (Curses.html#method-c-refresh) calls, returns `false` otherwise.

clrtoeol()

Clears to the end of line, that the cursor is currently on.

color_content(color)

Returns an 3 item Array of the RGB values in `color`

color_pair(attrs)

Sets the color pair attributes to `attrs`.

This should be equivalent to ::attrset (Curses.html#method-c-attrset)(COLOR_PAIR(`attrs`))

TODO: validate that equivalency

color_pairs()

Returns the COLOR_PAIRS available, if the curses library supports it.

colors()

returns COLORS (Curses.html#COLORS)

cols()

Returns the number of columns on the screen

crmode()

Put the terminal into normal mode (out of cbreak mode).

See ::cbreak (Curses.html#method-c-cbreak) for more detail.

curs_set(visibility)

Sets Cursor Visibility. 0: invisible 1: visible 2: very visible

def_prog_mode()

Save the current terminal modes as the "program" state for use by the ::reset_prog_mode (Curses.html#method-c-reset_prog_mode)

This is done automatically by ::init_screen (Curses.html#method-c-init_screen)

delch()

Delete the character under the cursor

deleteln()

Delete the line under the cursor.

doupdate()

Refreshes the windows and lines.

::doupdate (Curses.html#method-c-doupdate) allows multiple updates with more efficiency than ::refresh (Curses.html#method-c-refresh) alone.

echo()

Enables characters typed by the user to be echoed by ::getch (Curses.html#method-c-getch) as they are typed.

flash()

Flashs the screen, for visual alarm on the terminal, if possible; otherwise it sounds the alert.

see also ::beep (Curses.html#method-c-beep)

getch()

Read and returns a character from the window.

See Curses::Key (Curses/Key.html) to all the function KEY_* available

getmouse()

Returns coordinates of the mouse.

This will read and pop the mouse event data off the queue

See the BUTTON*, ALL_MOUSE_EVENTS (Curses.html#ALL_MOUSE_EVENTS) and REPORT_MOUSE_POSITION (Curses.html#REPORT_MOUSE_POSITION) constants, to examine the mask of the event

getstr()

This is equivalent to a series f Curses::Window#getch (Curses/Window.html#method-i-getch) calls

has_colors?()

Returns `true` or `false` depending on whether the terminal has color capbilities.

inch()

Returns the character at the current position.

init_color(color, r, g, b)

Changes the definition of a color. It takes four arguments:

- the number of the color to be changed, `color`
- the amount of red, `r`
- the amount of green, `g`
- the amount of blue, `b`

The value of the first argument must be between 0 and COLORS (Curses.html#COLORS). (See the section Colors for the default color index.) Each of the last three arguments must be a value between 0 and 1000. When ::init_color (Curses.html#method-c-init_color) is used, all occurrences of that color on the screen immediately change to the new definition.

init_pair(pair, f, b)

Changes the definition of a color-pair.

It takes three arguments: the number of the color-pair to be changed `pair`, the foreground color number `f`, and the background color number `b`.

If the color-pair was previously initialized, the screen is refreshed and all occurrences of that color-pair are changed to the new definition.

init_screen()

Initialize a standard screen

see also ::stdscr (Curses.html#method-c-stdscr)

insch(ch)

Insert a character `ch`, before the cursor.

insertln()

Inserts a line above the cursor, and the bottom line is lost

keyname(c)

Returns the character string corresponding to key `c`

lines()

Returns the number of lines on the screen

mouseinterval(interval)

The ::mouseinterval (Curses.html#method-c-mouseinterval) function sets the maximum time (in thousands of a second) that can elapse between press and release events for them to be recognized as a click.

Use ::mouseinterval (Curses.html#method-c-mouseinterval) to disable click resolution. This function returns the previous interval value.

Use ::mouseinterval (Curses.html#method-c-mouseinterval) to obtain the interval without altering it.

The default is one sixth of a second.

mousemask(mask)

Returns the `mask` of the reportable events

nl()

Enable the underlying display device to translate the return key into newline on input, and whether it translates newline into return and line-feed on output (in either case, the call ::addch (Curses.html#method-c-addch)('n') does the equivalent of return and line feed on the virtual screen).

Initially, these translations do occur. If you disable them using ::nonl (Curses.html#method-c-nonl), curses will be able to make better use of the line-feed capability, resulting in faster cursor motion. Also, curses will then be able to detect the return key.

nocbreak()

Put the terminal into normal mode (out of cbreak mode).

See ::cbreak (Curses.html#method-c-cbreak) for more detail.

nocrmode()

Put the terminal into normal mode (out of cbreak mode).

See ::cbreak (Curses.html#method-c-cbreak) for more detail.

noecho()

Disables characters typed by the user to be echoed by ::getch (Curses.html#method-c-getch) as they are typed.

nonl()

Disable the underlying display device to translate the return key into newline on input

See ::nl (Curses.html#method-c-nl) for more detail

noraw()

Put the terminal out of raw mode.

see ::raw (Curses.html#method-c-raw) for more detail

pair_content(pair)

Returns a 2 item Array, with the foreground and background color, in `pair`

pair_number(attrs)

Returns the Fixnum color pair number of attributes `attrs`.

raw()

Put the terminal into raw mode.

Raw mode is similar to ::cbreak (Curses.html#method-c-cbreak) mode, in that characters typed are immediately passed through to the user program.

The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal. The behavior of the BREAK key depends on other bits in the tty driver that are not set by curses.

refresh()

Refreshes the windows and lines.

reset_prog_mode()

Reset the current terminal modes to the saved state by the ::def_prog_mode (Curses.html#method-c-def_prog_mode)

This is done automatically by ::close_screen (Curses.html#method-c-close_screen)

resizeterm(lines, cols)

Resize the current term to Fixnum `lines` and Fixnum `cols`

resizeterm(lines, cols)

Resize the current term to Fixnum `lines` and Fixnum `cols`

scrl(num)

Scrolls the current window Fixnum `num` lines. The current cursor position is not changed.

For positive `num`, it scrolls up.

For negative `num`, it scrolls down.

setpos(y, x)

A setter for the position of the cursor, using coordinates `x` and `y`

setscrreg(top, bottom)

Set a software scrolling region in a window. `top` and `bottom` are lines numbers of the margin.

If this option and Curses.scrollok are enabled, an attempt to move off the bottom margin line causes all lines in the scrolling region to scroll one line in the direction of the first line. Only the text of the window is scrolled.

standend()

Enables the Normal display (no highlight)

This is equivalent to ::attron (Curses.html#method-c-attron)

see also Curses::Window#attrset (Curses/Window.html#method-i-attrset) for additional information.

standout()

Enables the best highlighting mode of the terminal.

This is equivalent to Curses:Curses::Window#attron (Curses/Window.html#method-i-attron)

see also Curses::Window#attrset (Curses/Window.html#method-i-attrset) additional information

start_color()

Initializes the color attributes, for terminals that support it.

This must be called, in order to use color attributes. It is good practice to call it just after ::init_screen (Curses.html#method-c-init_screen)

stdscr()

The Standard Screen.

Upon initializing curses, a default window called stdscr, which is the size of the terminal screen, is created.

Many curses functions use this window.

timeout=(delay)

Sets block and non-blocking reads for the window.

- If delay is negative, blocking read is used (i.e., waits indefinitely for input).
- If delay is zero, then non-blocking read is used (i.e., read returns ERR if no input is waiting).
- If delay is positive, then read blocks for delay milliseconds, and returns ERR if there is still no input.

ungetch(ch)

Places `ch` back onto the input queue to be returned by the next call to ::getch (Curses.html#method-c-getch).

There is just one input queue for all windows.

ungetmouse(p1)

It pushes a KEY_MOUSE (Curses.html#KEY_MOUSE) event onto the input queue, and associates with that event the given state data and screen-relative character-cell coordinates.

The ::ungetmouse (Curses.html#method-c-ungetmouse) function behaves analogously to ::ungetch (Curses.html#method-c-ungetch).

use_default_colors()

tells the curses library to use terminal's default colors.

see also the system manual for default_colors(3)

Commenting is here to help enhance the documentation. For example, sample code, or clarification of the documentation.

If you have questions about Ruby or the documentation, please post to one of the Ruby mailing lists (http://www.ruby-lang.org/en/community/mailing-lists/). You will get better, faster, help that way.

If you wish to post a correction of the docs, please do so, but also file bug report (http://bugs.ruby-lang.org/projects/ruby/wiki/HowtoReport) so that it can be corrected for the next release. Thank you.

Like

(#)

Add New Comment

Login (#)

Type your comment here.

Showing 0 comments

Sort by popular now

Subscribe by email (#) RSS (http://ruby-doc.disqus.com/thread_8145/latest.rss)

Trackback URL http://disqus.com/fo

blog comments powered by **DISQUS** (http://disqus.com)