Package ‘cursr’

January 11, 2021

Type  Package
Title  Cursor and Terminal Manipulation
Version  0.1.0
Author  Chris Mann
Maintainer  Chris Mann <cmann3@unl.edu>
Description  A toolbox for developing applications, games, simulations, or agent-based models in the R terminal. Included functions allow users to move the cursor around the terminal screen, change text colors and attributes, clear the screen, hide and show the cursor, map key presses to functions, draw shapes and curves, among others. Most functionalities require users to be in a terminal (not the R GUI).

Imports  keypress
Suggests  knitr, rmarkdown
VignetteBuilder  knitr
License  MIT + file LICENSE
Encoding  UTF-8
LazyData  true
RoxygenNote  7.0.2
NeedsCompilation  no
Repository  CRAN
Date/Publication  2021-01-11 08:50:09 UTC

R topics documented:

attr_off ................................................................. 3
attr_on ................................................................. 4
bg_off ................................................................. 5
bg_on ................................................................. 5
box_at ................................................................. 6
clear ................................................................. 7
color_off .............................................................. 8
color_pair ............................................................ 9
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>draw_arc</td>
<td>9</td>
</tr>
<tr>
<td>draw_bezier</td>
<td>10</td>
</tr>
<tr>
<td>draw_circle</td>
<td>11</td>
</tr>
<tr>
<td>draw_ellipse</td>
<td>12</td>
</tr>
<tr>
<td>draw_fn</td>
<td>13</td>
</tr>
<tr>
<td>draw_lerp</td>
<td>14</td>
</tr>
<tr>
<td>draw_path</td>
<td>14</td>
</tr>
<tr>
<td>draw_ray</td>
<td>15</td>
</tr>
<tr>
<td>draw_rect</td>
<td>16</td>
</tr>
<tr>
<td>draw_shape</td>
<td>17</td>
</tr>
<tr>
<td>erase</td>
<td>18</td>
</tr>
<tr>
<td>example_luckynumber</td>
<td>18</td>
</tr>
<tr>
<td>fg_off</td>
<td>19</td>
</tr>
<tr>
<td>fg_on</td>
<td>19</td>
</tr>
<tr>
<td>fill_circle</td>
<td>20</td>
</tr>
<tr>
<td>fill_ellipse</td>
<td>21</td>
</tr>
<tr>
<td>fill_rect</td>
<td>22</td>
</tr>
<tr>
<td>fill_shape</td>
<td>23</td>
</tr>
<tr>
<td>getkp</td>
<td>24</td>
</tr>
<tr>
<td>getkpl</td>
<td>24</td>
</tr>
<tr>
<td>grid_at</td>
<td>25</td>
</tr>
<tr>
<td>grid_mat</td>
<td>26</td>
</tr>
<tr>
<td>hide_cursor</td>
<td>27</td>
</tr>
<tr>
<td>hline</td>
<td>28</td>
</tr>
<tr>
<td>hline_at</td>
<td>28</td>
</tr>
<tr>
<td>in.term</td>
<td>29</td>
</tr>
<tr>
<td>load_cursor</td>
<td>30</td>
</tr>
<tr>
<td>make_bg</td>
<td>30</td>
</tr>
<tr>
<td>make_fg</td>
<td>31</td>
</tr>
<tr>
<td>make_style</td>
<td>32</td>
</tr>
<tr>
<td>mv</td>
<td>32</td>
</tr>
<tr>
<td>mv_col</td>
<td>33</td>
</tr>
<tr>
<td>mv_row</td>
<td>34</td>
</tr>
<tr>
<td>mv_to</td>
<td>35</td>
</tr>
<tr>
<td>path_arc</td>
<td>36</td>
</tr>
<tr>
<td>path_bezier</td>
<td>36</td>
</tr>
<tr>
<td>path_circle</td>
<td>37</td>
</tr>
<tr>
<td>path_ellipse</td>
<td>38</td>
</tr>
<tr>
<td>path_fill</td>
<td>39</td>
</tr>
<tr>
<td>path_fn</td>
<td>39</td>
</tr>
<tr>
<td>path_intersection</td>
<td>40</td>
</tr>
<tr>
<td>path_lerp</td>
<td>41</td>
</tr>
<tr>
<td>path_ray</td>
<td>41</td>
</tr>
<tr>
<td>path_rect</td>
<td>42</td>
</tr>
<tr>
<td>path_shape</td>
<td>43</td>
</tr>
<tr>
<td>repch</td>
<td>44</td>
</tr>
<tr>
<td>reset</td>
<td>44</td>
</tr>
<tr>
<td>save_cursor</td>
<td>45</td>
</tr>
</tbody>
</table>
**Description**

Turns off text attributes in the terminal, including bold text, italics, underline, etc.

**Usage**

`attr_off(...)`

**Arguments**

characters indicating attributes to turn off. "bf" for bold face; "ft" for faint; "it" for italics; "ul" for underline; "sb" for slow blink; "fb" for fast blink; "rv" for reverse video (invert bg and fg colors); "st" for strike-through. All attributes are turned off if left blank.

**Details**

Use `attr_on` to turn on attributes.

**Value**

`NULL`

**See Also**

Other style functions: `attr_on()`, `bg_off()`, `bg_on()`, `color_off()`, `color_pair()`, `fg_off()`, `fg_on()`, `make_bg()`, `make_fg()`, `make_style()`, `reset()`, `style()`
Examples

```r
cat("hello world!\n")
attr_on("bf", "ul")
cat("hello world!\n")
attr_off("bf")
cat("hello world!\n")
attr_off()
cat("hello world!\n")
```
bg_off

Turn Off Background Color

Description

Return the background of future terminal text to the default color. Background color is turned on with \texttt{bg\_on}.

Usage

\texttt{bg\_off()}

Value

\texttt{NULL}

See Also

Other style functions: \texttt{attr\_off()}, \texttt{attr\_on()}, \texttt{bg\_on()}, \texttt{color\_off()}, \texttt{color\_pair()}, \texttt{fg\_off()}, \texttt{fg\_on()}, \texttt{make\_bg()}, \texttt{make\_fg()}, \texttt{make\_style()}, \texttt{reset()}, \texttt{style()}

Examples

\begin{verbatim}
# Different methods of specifying yellow
bg_on("yellow")
bg_on("#FFFF00")
bg_on(11)
bg_on(255, 255, 0)

# Turn off color
bg_off()
\end{verbatim}

bg_on

Turn On Background Color

Description

Specifies the background color of all future text written in the terminal \texttt{bg\_on} accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color. Not all terminals support each possible color.

Usage

\texttt{bg\_on(...)}

Arguments

... character or numeric value

Details

Background color is turned off with bg_off.

Value

NULL

See Also

Other style functions: attr_off(), attr_on(), bg_off(), color_off(), color_pair(), fg_off(),
fg_on(), make_bg(), make_fg(), make_style(), reset(), style()

Examples

# Different methods of specifying yellow
bg_on("yellow")
bg_on("#FFFF00")
bg_on(11)
bg_on(255, 255, 0)

# Turn off color
bg_off()

box_at

box_at

Draw Box

Description

Draws a box of size dim=c(height, width) at yx=c(row, col).

Usage

box_at(
    yx = c(1, 1),
    dim = NULL,
    text = c("|", "|", "-", "-", rep("+", 4)),
    fg = NA,
    bg = NA,
    attr = NA,
    fill = NA,
    fill.bg = NA,
    fill.fg = NA,
    fill.attr = NA
)
Clear text from the terminal. Passing values "start" or "end" allow the user to clear specific portions of the screen relative to the cursor.

clear(x = c("screen", "end", "start"), ...)
Arguments

`x` character describing console location to clear. The default, "screen", clears the entire screen; "start" clears all text from the beginning of the screen until the cursor's position; "end" clears all text from the cursor's position to the bottom of the screen.

Value

NULL

Examples

clear()
cat(paste(LETTERS[1:10], collapse="\n"))
clear("start")
clear("end")

color_off

Turn Off Colors in Terminal

Description

Return the background and foreground of future terminal text to the default colors.

Usage

color_off()

Value

NULL

See Also

Other style functions: `attr_off()`, `attr_on()`, `bg_off()`, `bg_on()`, `color_pair()`, `fg_off()`, `fg_on()`, `make_bg()`, `make_fg()`, `make_style()`, `reset()`, `style()`

Examples

bg_on("red")
fg_on("yellow")

# Turn off color
color_off()
**color_pair**

Create Background & Foreground Color Combination

**Description**

Returns the ANSI codes for the specified colors. color_pair accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color.

**Usage**

color_pair(fg, bg)

**Arguments**

- **fg**: character or numeric value for the foreground color
- **bg**: character or numeric value for the background color

**Value**

ANSI character string

**See Also**

Other style functions: attr_off(), attr_on(), bg_off(), bg_on(), color_off(), fg_off(), fg_on(), make_bg(), make_fg(), make_style(), reset(), style()

**Examples**

- # Blue background with white text
  - color_pair("white", "blue")
  - color_pair("#FFFFFF", "#0000FF")
  - color_pair(0, 12)
  - color_pair(c(255, 255, 255), c(0,0,255))

**draw_arc**

Draw an Arc

**Description**

Calculate the path of an arc within a grid and print to screen.

**Usage**

draw_arc(yx, start, end, r = 1, n = 50, text = "x", ...)

Arguments

- **yx**  center (row, col) coordinate of circle
- **start**  starting angle in radians
- **end**  ending angle in radians
- **r**  radius of circle
- **n**  number of points along curve to calculate
- **text**  character value drawn at coordinate
- **...**  parameters that are passed to `style()`, including the foreground color `fg`, background color `bg`, and attribute `attr`

Value

NULL

See Also

Other drawing functions: `box_at()`, `draw_bezier()`, `draw_circle()`, `draw_ellipse()`, `draw_fn()`, `draw_lerp()`, `draw_path()`, `draw_ray()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `fill_shape()`, `grid_at()`, `grid_mat()`, `hline_at()`, `hline()`, `vline_at()`, `vline()`

Examples

```r
draw_arc(yx=c(10,10), start=pi/2, end=pi, r=6)
```

---

**draw_bezier**  
**Draw a Bezier Curve**

Description

Calculate the path of a Bezier Curve with up to two control points in a grid and draw to screen.

Usage

```r
draw_bezier(start, end, c1, c2 = NULL, n = 50, text = "x", ...)
```

Arguments

- **start**  starting (row, col) coordinate
- **end**  ending (row, col) coordinate
- **c1**  coordinate of first control point
- **c2**  coordinate of second control point
- **n**  number of points along curve to calculate
- **text**  character value drawn at coordinate
- **...**  parameters that are passed to `style()`, including the foreground color `fg`, background color `bg`, and attribute `attr`
draw_circle

Value

NULL

See Also

Other drawing functions: box_at(), draw_arc(), draw_bezier(), draw_ellipse(), draw_fn(), draw_lerp(), draw_path(), draw_ray(), draw_rect(), draw_shape(), fill_circle(), fill_ellipse(), fill_rect(), fill_shape(), grid_at(), grid_mat(), hline_at(), hline(), vline_at(), vline()

Examples

draw_bezier(start=c(10,1), end=c(10,10), c1=c(1,3))

draw_circle(yx=c(10,10), r=5)
draw_ellipse  

**Draw Ellipse**

**Description**

Calculate the path of an ellipse within a grid and draw to screen.

**Usage**

```r
draw_ellipse(yx = c(0, 0), rx = 1, ry = 1, n = 50, text = "x", ...)
```

**Arguments**

- `yx` (row, col) coordinate of the center of the ellipse
- `rx` radius along the x-axis in grid points
- `ry` radius along the y-axis in grid points
- `n` number of points along curve to calculate
- `text` character value drawn at coordinate
- `...` parameters that are passed to `style()`, including the foreground color `fg`, background color `bg`, and attribute `attr`

**Value**

`NULL`

**See Also**

Other drawing functions: `box_at()`, `draw_arc()`, `draw_bezier()`, `draw_circle()`, `draw_fn()`, `draw_lerp()`, `draw_path()`, `draw_ray()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `fill_shape()`, `grid_at()`, `grid_mat()`, `hline_at()`, `hline()`, `vline_at()`, `vline()`

**Examples**

```r
draw_ellipse(yx=c(10,10), rx=8, ry = 4)
```
**draw_fn**  

**Draw a Function**

**Description**

Calculate the path within a grid of an user-supplied function and print to screen.

**Usage**

```r
draw_fn(x1, x2, fn, n = 50, text = "x", ...)```

**Arguments**

- `x1` starting column value of the path
- `x2` ending column value of the path
- `fn` function returning row value for a column input
- `n` number of points along curve to calculate
- `text` character value drawn at coordinate
- `...` parameters that are passed to `style()`, including the foreground color `fg`, background color `bg`, and attribute `attr`

**Value**

`NULL`

**See Also**

Other drawing functions: `box_at()`, `draw_arc()`, `draw_bezier()`, `draw_circle()`, `draw_ellipse()`, `draw_lerp()`, `draw_path()`, `draw_ray()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `fill_shape()`, `grid_at()`, `grid_mat()`, `hline_at()`, `hline()`, `vline_at()`, `vline()`

**Examples**

```r
draw_fn(x1=1, x2=10,  
      function(x){sqrt(x)})
```
**draw_lerp**  

*Draw a Line*

**Description**

Interpolate between two points in a grid and draw to screen.

**Usage**

```r
draw_lerp(start, end, n = 50, text = "x", ...)
```

**Arguments**

- `start`: starting (row, col) coordinate
- `end`: ending (row, col) coordinate
- `n`: number of points along curve to calculate
- `text`: character value drawn at coordinate
- `...`: parameters that are passed to `style()`, including the foreground color `fg`, background color `bg`, and attribute `attr`

**Value**

`NULL`

**See Also**

Other drawing functions: `box_at()`, `draw_arc()`, `draw_bezier()`, `draw_circle()`, `draw_ellipse()`, `draw_fn()`, `draw_path()`, `draw_ray()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `fill_shape()`, `grid_at()`, `grid_mat()`, `hline_at()`, `hline()`, `vline_at()`, `vline()`

**Examples**

```r
draw_lerp(start=c(10,1), end=c(1,3))
```

---

**draw_path**  

*Draw Path*

**Description**

Draws text at each supplied coordinate.

**Usage**

```r
draw_path(coord, text = "x", ...)
```
**draw_ray**

**Arguments**
- **coord**: matrix or list containing (row, col) coordinates.
- **text**: character value drawn at coordinate
- ... parameters that are passed to style(), including the foreground color fg, background color bg, and attribute attr

**Value**
NULL

**See Also**
Other drawing functions: box_at(), draw_arc(), draw_bezier(), draw_circle(), draw_ellipse(), draw_fn(), draw_lerp(), draw_ray(), draw_rect(), draw_shape(), fill_circle(), fill_ellipse(), fill_rect(), fill_shape(), grid_at(), grid_mat(), hline_at(), hline(), vline_at(), vline()

**Examples**
```r
c <- path_circle(yx = c(5, 5), r=3)
draw_path(c, text="0")
```

---

**draw_ray**

**Description**
Calculate the path of a ray extending and print to screen.

**Usage**
```r
draw_ray(start, end, lim = c(64, 128), n = 200, text = "x", ...)
```

**Arguments**
- **start**: start (row, col) coordinate of the ray
- **end**: either an ending coordinate, an angle in radians, or a character direction (u, d, l, r, ul, ur, dl, dr)
- **lim**: bounding box dimensions used to calculate ray
- **n**: number of points along curve to calculate
- **text**: character value drawn at coordinate
- ... parameters that are passed to style(), including the foreground color fg, background color bg, and attribute attr
**Value**

NULL

**See Also**

Other drawing functions: `box_at()`, `draw_arc()`, `draw_bezier()`, `draw_circle()`, `draw_ellipse()`, `draw_fn()`, `draw_lerp()`, `draw_path()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `fill_shape()`, `grid_at()`, `grid_mat()`, `hline_at()`, `hline()`, `vline_at()`, `vline()`

**Examples**

```r
draw_rect(c(5,5), c(9,9))
```
**draw_shape**

---

**Draw a Shape**

---

**Description**

Calculate the path of a shape given supplied vertices and draw to screen.

**Usage**

draw_shape(mat, cycle = TRUE, n = 30, text = "x", ...)

**Arguments**

- **mat**: an Nx2 matrix of (row,col) coordinates
- **cycle**: logical value determining whether to the first and last coordinates
- **n**: number of points along each edge to calculate
- **text**: character value drawn at coordinate
- **...**: parameters that are passed to style(), including the foreground color fg, background color bg, and attribute attr

**Value**

NULL

**See Also**

Other drawing functions: box_at(), draw_arc(), draw_bezier(), draw_circle(), draw_ellipse(), draw_fn(), draw_lerp(), draw_path(), draw_ray(), draw_rect(), fill_circle(), fill_ellipse(), fill_rect(), fill_shape(), grid_at(), grid_mat(), hline_at(), hline(), vline_at(), vline()

**Examples**

```r
# Right Triangle
draw_shape(rbind(
c(10,1),
c(10,10),
c(1,1)
), cycle=TRUE)
```
erase  

Erase Text

Description
Clear text from the cursor’s row. Passing values "start" and "end" allow the user to erase specific portions of the row relative to the cursor.

Usage
erase(x = c("row", "start", "end"), ...)

Arguments
x  character describing location to clear. The default, "row", clears the entire row; "start" clears all text from the beginning of the row until the cursor’s position; "end" clears all text from the cursor’s position until the end of the row.
...
objects passed to/from methods

Value
NULL

Examples
   cat("hello world!")
   erase('row')


display_data  

Example Program From Vignette

Description
Simple program that asks for a letter and a number and returns another value to screen.

Usage
display_data()

Value
NULL
**fg_off**

*Turn Off Foreground Color*

**Description**

Return future terminal text to the default color. Foreground color is turned on with `fg_on`.

**Usage**

```c
fg_off()
```

**Value**

NULL

**See Also**

Other style functions: `attr_off()`, `attr_on()`, `bg_off()`, `bg_on()`, `color_off()`, `color_pair()`, `fg_on()`, `make_bg()`, `make_fg()`, `make_style()`, `reset()`, `style()`

**Examples**

```c
# Different methods of specifying red
fg_on("red")
fg_on("#FF0000")
fg_on(1)
fg_on(255, 0, 0)

# Turn off color
fg_off()
```

---

**fg_on**

*Turn On Foreground Color*

**Description**

Specifies the color of all future text written in the terminal `fg_on` accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color. Not all terminals support each possible color.

**Usage**

```c
fg_on(...)```

---
Arguments

... character or numeric value

Details

Foreground color is turned off with `fg_off`.

Value

`NULL`

See Also

Other style functions: `attr_off()`, `attr_on()`, `bg_off()`, `bg_on()`, `color_off()`, `color_pair()`, `fg_off()`, `make_bg()`, `make_fg()`, `make_style()`, `reset()`, `style()`

Examples

# Different methods of specifying red
fg_on("red")
fg_on("#FF0000")
fg_on(9)
fg_on(255, 0, 0)

# Turn off color
fg_off()
\textit{fill_ellipse}

\textbf{Value}

\texttt{NULL}

\textbf{See Also}

Other drawing functions: \texttt{box_at()}, \texttt{draw_arc()}, \texttt{draw bezier()}, \texttt{draw_circle()}, \texttt{draw ellipse()}, \texttt{draw fn()}, \texttt{draw lerp()}, \texttt{draw path()}, \texttt{draw ray()}, \texttt{draw rect()}, \texttt{draw shape()}, \texttt{fill ellipse()}, \texttt{fill rect()}, \texttt{fill shape()}, \texttt{grid at()}, \texttt{grid mat()}, \texttt{hline at()}, \texttt{hline()}, \texttt{vline at()}, \texttt{vline()}

\textbf{Examples}

draw_circle(\texttt{yx=c(10,10), r=5})

---

\textit{fill_ellipse} \hspace{1cm} \textit{Draw a Filled-In Ellipse}

\textbf{Description}

Calculate the path of an ellipse within a grid and draw to screen.

\textbf{Usage}

\texttt{fill_ellipse(\texttt{yx = c(0, 0), rx = 1, ry = 1, n = 50, text = "x", ...})}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{yx} \hspace{1cm} \text{(row,col)} coordinate of the center of the ellipse
  \item \texttt{rx} \hspace{1cm} radius along the x-axis in grid points
  \item \texttt{ry} \hspace{1cm} radius along the y-axis in grid points
  \item \texttt{n} \hspace{1cm} number of points along curve to calculate
  \item \texttt{text} \hspace{1cm} character value drawn at coordinate
  \item \ldots \hspace{1cm} parameters that are passed to \texttt{style()}, including the foreground color \texttt{fg}, background color \texttt{bg}, and attribute \texttt{attr}
\end{itemize}

\textbf{Value}

\texttt{NULL}

\textbf{See Also}

Other drawing functions: \texttt{box at()}, \texttt{draw arc()}, \texttt{draw bezier()}, \texttt{draw circle()}, \texttt{draw ellipse()}, \texttt{draw fn()}, \texttt{draw lerp()}, \texttt{draw path()}, \texttt{draw ray()}, \texttt{draw rect()}, \texttt{draw shape()}, \texttt{fill ellipse()}, \texttt{fill rect()}, \texttt{fill shape()}, \texttt{grid at()}, \texttt{grid mat()}, \texttt{hline at()}, \texttt{hline()}, \texttt{vline at()}, \texttt{vline()}
Examples

draw_ellipse(yx=c(10,10), rx=8, ry = 4)

draw_rect(c(5,5), c(9,9))
**fill_shape**

**Draw a Filled-In Shape**

**Description**

Calculate the path of a shape given supplied vertices and draw to screen.

**Usage**

```r
fill_shape(mat, cycle = TRUE, n = 30, text = "x", ...)  
```

**Arguments**

- **mat**: an Nx2 matrix of (row, col) coordinates
- **cycle**: logical value determining whether to the first and last coordinates
- **n**: number of points along each edge to calculate
- **text**: character value drawn at coordinate
- **...**: parameters that are passed to `style()`, including the foreground color `fg`, background color `bg`, and attribute `attr`

**Value**

- `NULL`

**See Also**

Other drawing functions: `box_at()`, `draw_arc()`, `draw_bezier()`, `draw_circle()`, `draw_ellipse()`, `draw_fn()`, `draw_lerp()`, `draw_path()`, `draw_ray()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `grid_at()`, `grid_mat()`, `hline_at()`, `hline()`, `vline_at()`, `vline()`

**Examples**

```r
# Right Triangle  
fill_shape(rbind(  
  c(10,1),  
  c(10,10),  
  c(1,1)  
), cycle=TRUE)
```
**getkp**  
Get Keypress

**Description**

Listen for a keypress, then apply keypress to a function or echo it to the terminal screen. The user must be in a terminal to use `getkp`; it will not work in RStudio or the R GUI. All actions within R are halted until the keypress is returned.

**Usage**

```r
getkp(fn = list(), echo = FALSE)
```

**Arguments**

- `fn` list of named functions  
- `echo` whether the keypress should be echoed to the screen if not found in list

**Value**

character naming the key that was pressed (invisibly).

**Examples**

```r
f <- list(
  'up' = function(){mv(row=-1)},
  'down' = function(){mv(row=1)},
  'left' = function(){mv(col=-1)},
  'right' = function(){mv(col=1)}
)
getkp(fn=f, echo=FALSE)
```

**getkpl**  
Loop a Keypress

**Description**

Maintain a loop that listens for a keypress, then applies the keypress to a function or echoes it to the terminal screen. The user must be in a terminal to use `getkp`; it will not work in RStudio or the R GUI. All actions within R are halted until the keypress is returned.
Usage

getkpl(escape = "escape", fn = list(), echo = FALSE)

Arguments

escape vector of character keypresses that escape the loop. The default is "escape" key.
fn list of named functions
echo whether the keypress should be echoed to the screen if not found in list

Value

NULL

Examples

f <- list(
  'up' = function(){mv(row=-1)},
  'down' = function(){mv(row=-1)},
  'left' = function(){mv(col=-1)},
  'right' = function(){mv(col=1)}
)
## Not run:
getkpl(escape = c("escape", "enter"), fn=f, echo=FALSE)

## End(Not run)

grid_at

Draw a Character Grid Matrix

Description

Constructs a grid with given dimension, character values, and step parameter, and prints it to screen

Usage

grid_at(
  yx = c(1, 1),
  dim = NULL,
  step = c(2, 2),
  text = c(".", ".", "+", ":", ":", ":", ":", ":", rep("+", 8)),
  border = TRUE
)
Arguments

- **yx**: \((row, column)\) on screen or window where the upper-left corner of the grid is to be printed.
- **dim**: \((row, column)\) vector for size of grid.
- **step**: numeric vector describing grid step across \((rows, columns)\).
- **text**: character vector of values for the grid, in order: horizontal grid line, vertical grid line, grid intersection, left border, right border, top border, bottom border, corners (upper-left, upper-right, lower-left, lower-right), ticks (right, bottom, left, top).
- **border**: logical value for whether a border should be included.

Value

NULL

See Also

Other drawing functions: `box_at()`, `draw_arc()`, `draw_bezier()`, `draw_circle()`, `draw_ellipse()`, `draw_fn()`, `draw_lerp()`, `draw_path()`, `draw_ray()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `fill_shape()`, `grid_mat()`, `hline_at()`, `hline()`, `vline_at()`, `vline()`

Examples

```r
grid_at(yx=c(2,2), dim=c(11,13), step=c(2,4), border=TRUE)
```

---

**grid_mat**  
Create a Character Grid Matrix

Description

Constructs a grid with provided dimensions \((row, col)\), character values for gridlines, and a step parameter noting the number of rows and columns between each gridline.

Usage

```r
grid_mat(
  dim,
  step = c(2, 2),
  text = c("\.", ".", "+", "|", "|", "-", "-", rep("+", 8)),
  border = TRUE
)
```
**Arguments**

- **dim** (row, column) vector for size of grid.
- **step** numeric vector describing grid step across (rows, columns).
- **text** character vector of values for the grid, in order: horizontal grid line, vertical grid line, grid intersection, left border, right border, top border, bottom border, corners (upper-left, upper-right, lower-left, lower-right), ticks (right, bottom, left, top).
- **border** logical value for whether a border should be included.

**Value**

rowxcol matrix

**See Also**

Other drawing functions: box_at(), draw_arc(), draw_bezier(), draw_circle(), draw_ellipse(), draw_fn(), draw_lerp(), draw_path(), draw_ray(), draw_rect(), draw_shape(), fill_circle(), fill_ellipse(), fill_rect(), fill_shape(), grid_at(), hline_at(), hline(), vline_at(), vline()

**Examples**

```r
grid_mat(dim=c(11,13), step=c(2,4), border=TRUE)
```

---

**Description**

Make the cursor invisible. The cursor can be revealed with `show_cursor`

**Usage**

`hide_cursor()`

**Value**

`NULL`

**Examples**

```r
hide_cursor()
show_cursor()
```
**hline**

*Horizontal Line*

**Description**

Horizontal Line

**Usage**

\[
\text{hline}(n, \text{text} = "-")
\]

**Arguments**

- **n**  
  integer describing the character length of the line
- **text**  
  character to be repeated

**Value**

character string of length \( n \)

**See Also**

Other drawing functions: `box_at()`, `draw_arc()`, `draw_bezier()`, `draw_circle()`, `draw_ellipse()`, `draw_fn()`, `draw_lerp()`, `draw_path()`, `draw_ray()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `fill_shape()`, `grid_at()`, `grid_mat()`, `hline_at()`, `vline_at()`, `vline()`

**Examples**

```
hline(10, "*")  # **********
hline(5, "$")   # $$$$$$
```

---

**hline_at**

*Draw Horizontal Line*

**Description**

Draws a horizontal line of length \( n \) at \((\text{row}, \text{col})\)

**Usage**

\[
\text{hline_at}(\text{yx}, n, \text{text} = "-", \ldots)
\]
Arguments

- **yx** (row,col) coordinates where line should be drawn.
- **n** integer describing the character length of the line
- **text** character to be repeated
- **...** parameters that are passed to `style()`, including the foreground color `fg`, background color `bg`, and attribute `attr`.

Value

NULL

See Also

Other drawing functions: `box_at()`, `draw_arc()`, `draw_bezier()`, `draw_circle()`, `draw_ellipse()`, `draw_fn()`, `draw_lerp()`, `draw_path()`, `draw_ray()`, `draw_rect()`, `draw_shape()`, `fill_circle()`, `fill_ellipse()`, `fill_rect()`, `fill_shape()`, `grid_at()`, `grid_mat()`, `hline()`, `vline_at()`, `vline()`.

Examples

```r
hline_at(c(3,4),6,"-") # print "-------" at (3,4)
```

---

### in.term

**Determine whether in Terminal**

Description

Tests whether the session is in terminal and returns TRUE or FALSE. Many of the `cursor` functions require being in terminal and will not work with RStudio or the R GUI application.

Usage

```r
in.term()
```

Value

logical value; TRUE or FALSE

Examples

```r
in.term()
```
load_cursor

Load Cursor

Description
Restore cursor to its previously saved location from save_cursor.

Usage
load_cursor()

Value
NULL

Examples
save_cursor()
cat("\n\nHello World!")
load_cursor()

make_bg

Create Background Color

Description
Returns the ANSI code for the specified background color. make_bg accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color.

Usage
make_bg(...)

Arguments
... character or numeric value

Value
ANSI character string

See Also
Other style functions: attr_off(), attr_on(), bg_off(), bg_on(), color_off(), color_pair(), fg_off(), fg_on(), make_fg(), make_style(), reset(), style()
make_fg

Examples

# Different methods of specifying cyan
make_bg("cyan")
make_bg("#00FFFF")
make_bg(14)
make_bg(0, 255, 255)

---

make_fg

Create Foreground Color

Description

Returns the ANSI code for the specified foreground color. make_fg accepts numeric values (RGB or 8-bit color code), hexadecimal characters, or the name of the color.

Usage

make_fg(...)  

Arguments

... character or numeric value

Value

ANSI character string

See Also

Other style functions: attr_off(), attr_on(), bg_off(), bg_on(), color_off(), color_pair(), fg_off(), fg_on(), make_bg(), make_style(), reset(), style()

Examples

# Different methods of specifying red
make_fg("red")
make_fg("#FF0000")
make_fg(9)
make_fg(255, 0, 0)
make_style  
Create Color & Attribute Style

Description

Returns the ANSI codes for the specified colors and text attributes.

Usage

make_style(fg = NA, bg = NA, attr = NA)

Arguments

fg  character or numeric value for the foreground color. See fg_on for more details.
bg  character or numeric value for the background color. See bg_on for more details.
attr character vector describing attributes to turn on. See attr_on for more details.

Value

ANSI character string

See Also

Other style functions: attr_off(), attr_on(), bg_off(), bg_on(), color_off(), color_pair(),
fg_off(), fg_on(), make_bg(), make_fg(), reset(), style()

Examples

cat(make_style(fg="blue", bg=c(192,192,192), attr=c("ul", "st")))
cat("Hello World!\n")
reset()

mv  
Move Cursor

Description

Move cursor relative to its current position on the screen. Screen coordinates are given by (row, column) with the position of the screen being (1,1).

Usage

mv(row = 0L, col = 0L)
Arguments

row  
number of rows in which to move the cursor. Positive values move the cursor down; negative values move the cursor up. If row has two or more values, the second value replaces col.

col  
number of columns in which to move the cursor. Positive values move the cursor forward; negative values move the cursor backwards.

Details

The user must be in a terminal to use the functionality; it will not work in RStudio or the R GUI.

Value

NULL

See Also

mv_to to move to a specific location on the screen.

Other moving functions: mv_col(), mv_row(), mv_to()

Examples

  # move the cursor down one and forward two
  mv(1, 2)

  # Alternatively, you can specify the coordinates as a single vector.
  loc <- c(1, 2)
  mv(loc)

  # to move to the left one unit (only works if the current column is > 1)
  mv(, -1)

---

mv_col  Move Cursor to Column

Description

Move the cursor to the specified column, while maintaining the same row.

Usage

mv_col(n = 1L)

Arguments

n  
positive integer specifying the column
Details
The user must be in a terminal to use the functionality; it will not work in RStudio or the R GUI.

Value
NULL

See Also
Other moving functions: mv_row(), mv_to(), mv()

Examples

# Position cursor at the beginning of the row
mv_col(1)

# Move cursor to the 10th column in the row
mv_col(10)
mv_to

Examples

# move the cursor to the beginning of the previous line
mv_row(-1)

Description

Move cursor relative to its current position on the screen. Screen coordinates are given by \((row, column)\) with the position of the screen being \((1,1)\).

Usage

mv_to(row = 1L, col = 1L)

Arguments

row positive integer specifying the console row. If row has two or more values, the second value replaces col.

col positive integer specifying the console column.

Details

The user must be in a terminal to use the functionality; it will not work in RStudio or the R GUI.

Value

NULL

See Also

mv to move relative to the current location on the screen.

Other moving functions: \(\text{mv\_col()}, \text{mv\_row()}, \text{mv}\)

Examples

# move the cursor to the 2nd row, 4th column
mv_to(2, 4)

# alternatively, you can specify the coordinates as a vector.
loc <- c(2, 4)
mv_to(loc)
**path_arc**  
*Arc Path*

**Description**
Calculate the path of an arc within a grid.

**Usage**
opath_arc(yx, start, end, r = 1, n = 50)

**Arguments**
- **yx**: center (row, col) coordinate of circle
- **start**: starting angle in radians
- **end**: ending angle in radians
- **r**: radius of circle
- **n**: number of points along curve to calculate

**Value**
Nx2 matrix of (row, column) coordinates

**See Also**
Other path-fitting functions: path_bezier(), path_circle(), path_ellipse(), path_fill(), path_fn(), path_intersection(), path_lerp(), path_ray(), path_rect(), path_shape()

**Examples**
path_arc(yx=c(10,10), start=pi/2, end=pi, r=6)

---

**path_bezier**  
*Bezier Curve Path*

**Description**
Calculate the path of a Bezier Curve with up to two control points in a grid.

**Usage**
opath_bezier(start, end, c1, c2 = NULL, n = 50)

**Examples**
path_bezier(start, end, c1, c2 = NULL, n = 50)
path_circle

Arguments

- **start**: starting (row, col) coordinate
- **end**: ending (row, col) coordinate
- **c1**: coordinate of first control point
- **c2**: coordinate of second control point
- **n**: number of points along curve to calculate

Value

Nx2 matrix of (row, column) coordinates

See Also

Other path-fitting functions: path_arc(), path_bezier(), path_ellipse(), path_fill(), path_fn(), path_intersection(), path_lerp(), path_ray(), path_rect(), path_shape()

Examples

```r
path_bezier(start = c(10, 1), end = c(10, 10), c1 = c(1, 3))
```

---

**Description**

Calculate the path of a circle in a grid.

**Usage**

```r
path_circle(yx, r = 1, n = 50)
```

**Arguments**

- **yx**: center (row, col) coordinate
- **r**: radius of the circle in grid points
- **n**: number of points along curve to calculate

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: path_arc(), path_bezier(), path_ellipse(), path_fill(), path_fn(), path_intersection(), path_lerp(), path_ray(), path_rect(), path_shape()
Examples

```r
path_circle(yx=c(10,10), r=5)
```

---

**path_ellipse**  
**Ellipse Path**

Description

Calculate the path of an ellipse within a grid.

Usage

```r
path_ellipse(yx = c(0, 0), rx = 1, ry = 1, n = 50)
```

Arguments

- `yx` (row, col) coordinate of the center of the ellipse
- `rx` radius along the x-axis in grid points
- `ry` radius along the y-axis in grid points
- `n` number of points along curve to calculate

Value

Nx2 matrix of (row, column) coordinates

See Also

Other path-fitting functions: `path_arc()`, `path_bezier()`, `path_circle()`, `path_fill()`, `path_fn()`, `path_intersection()`, `path_lerp()`, `path_ray()`, `path_rect()`, `path_shape()`

Examples

```r
path_ellipse(yx=c(10,10), rx=8, ry = 4)
```
**path_fill**  

*Fill In Path*

**Description**

Calculate the coordinates of all points inside of a path.

**Usage**

```
path_fill(mat)
```

**Arguments**

- **mat**: Nx2 matrix of (row, column) path coordinates

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions:  
- `path_arc()`  
- `path_bezier()`  
- `path_circle()`  
- `path_ellipse()`  
- `path_fn()`  
- `path_intersection()`  
- `path_lerp()`  
- `path_rect()`  
- `path_shape()`

**Examples**

```r
 investigates <- path_circle(c(10,10), r=5)
 path_fill(c0)
```

---

**path_fn**  

*Function Path*

**Description**

Calculate the path within a grid of an user-supplied function.

**Usage**

```
path_fn(x1, x2, fn, n = 50)
```

**Arguments**

- **x1**: starting column value of the path  
- **x2**: ending column value of the path  
- **fn**: function returning row value for a column input  
- **n**: number of points along curve to calculate
Value

Nx2 matrix of (row, column) coordinates

See Also

Other path-fitting functions: path_arc(), path_bezier(), path_circle(), path_ellipse(), path_fill(), path_intersection(), path_lerp(), path_ray(), path_rect(), path_shape()

Examples

c1 <- path_circle(c(4,4), r=3)
c2 <- path_circle(c(6,6), r=3)
path_intersection(list(c1, c2))
path_lerp  

Linear Interpolation Path

Description
Interpolate between two points in a grid.

Usage
path_lerp(start, end, n = 50)

Arguments
- start: starting (row, col) coordinate
- end: ending (row, col) coordinate
- n: number of points along curve to calculate

Value
Nx2 matrix of (row, column) coordinates

See Also
Other path-fitting functions: path_arc(), path_bezier(), path_circle(), path_ellipse(), path_fill(), path_fn(), path_intersection(), path_ray(), path_rect(), path_shape()

Examples
path_lerp(start=c(10,1), end=c(1,3))

path_ray  

Ray Path

Description
Calculate the path of a ray extending

Usage
path_ray(start, end, lim = c(64, 128), n = 200)
**path_rect**

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>start (row, col) coordinate of the ray</td>
</tr>
<tr>
<td>end</td>
<td>either an ending coordinate, an angle in radians, or a character direction (u, d, l, r, ul, ur, dl, dr)</td>
</tr>
<tr>
<td>lim</td>
<td>bounding box dimensions used to calculate ray</td>
</tr>
<tr>
<td>n</td>
<td>number of points along curve to calculate</td>
</tr>
</tbody>
</table>

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: path_arc(), path_bezier(), path_circle(), path_ellipse(), path_fill(), path_fn(), path_intersection(), path_lerp(), path_rect(), path_shape()

**Examples**

```r
path(ray(start=c(10,10), end=pi/6))
path(ray(start=c(10,10), end=pi/6, lim=c(15,15)))
path(ray(start=c(10,10), end=c(4,2)))
```

---

**path_rect**  
*Rectangle Path*

**Description**

Calculate the path of a rectangle in a grid.

**Usage**

```r
path_rect(yx1, yx2)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yx1</td>
<td>upper-left (row, col) coordinate</td>
</tr>
<tr>
<td>yx2</td>
<td>lower-right (row, col) coordinate</td>
</tr>
</tbody>
</table>

**Value**

Nx2 matrix of (row, column) coordinates

**See Also**

Other path-fitting functions: path_arc(), path_bezier(), path_circle(), path_ellipse(), path_fill(), path_fn(), path_intersection(), path_lerp(), path_ray(), path_shape()
path_shape

Examples

path_rect(c(5,5), c(9,9))

path_shape(mat, cycle = TRUE, n = 30)

Arguments

mat an Nx2 matrix of (row,col) coordinates

cycle logical value determining whether to the first and last coordinates

n number of points along each edge to calculate

Value

Nx2 matrix of (row,column) coordinates

See Also

Other path-fitting functions: path_arc(), path_bezier(), path_circle(), path_ellipse(), path_fill(), path_fn(), path_intersection(), path_around(), path_rect()

Examples

# Right Triangle
path_shape(rbind(
  c(10,1),
  c(10,10),
  c(1,1)
), cycle=TRUE)
**repch**  
*Repeat a Character*

**Description**

Repeat a character \( n \) times and concatenate into a single value.

**Usage**

\[
\text{repch}(x, n)
\]

**Arguments**

- \( x \)  
  character to be repeated
- \( n \)  
  number of times to be repeated

**Value**

character vector

**Examples**

\[
\text{repch(“abc”, 5)}
\]

---

**reset**  
*Reset Console Style*

**Description**

Turns off all text attributes and colors in the terminal.

**Usage**

\[
\text{reset()}
\]

**Value**

NULL

**See Also**

Other style functions: `attr_off()`, `attr_on()`, `bg_off()`, `bg_on()`, `color_off()`, `color_pair()`, `fg_off()`, `fg_on()`, `make_bg()`, `make_fg()`, `make_style()`, `style()`
Examples

attr_on("ul")
fg_on("red")
bg_on(c(10,60,205))
cat("Hello World!\n")

reset()
cat("Hello World!\n")

---

**save_cursor**  
**Save Cursor Position**

Description

Save the position of the cursor so that the position can be restored for later with load_cursor.

Usage

```r
save_cursor()
```

Value

NULL

Examples

```r
save_cursor()
cat("\n\nHello World!")
load_cursor()
```

---

**show_cursor**  
**Show Cursor**

Description

Reveal the cursor after it has been hidden by hide_cursor.

Usage

```r
show_cursor()
```

Value

NULL
Examples

```r
hide_cursor()
cat("\n\nHello World!")
show_cursor()
```

**style**  
*Add Color & Attributes to a Character*

**Description**
Add color and other text attributes to a character vector. Attributes can be seen after text is passed to `cat`, though it may only show up in a terminal. Note that terminal attributes and colors are automatically reset to default after text is printed.

**Usage**
```r
style(x, fg = NA, bg = NA, attr = NA)
```

**Arguments**
- `x`: character vector to be styled
- `fg`: character or numeric value for the foreground color. See `fg_on` for more details.
- `bg`: character or numeric value for the background color. See `bg_on` for more details.
- `attr`: character vector describing attributes to turn on. See `attr_on` for more details.

**Value**
character vector

**See Also**
Other style functions: `attr_off()`, `attr_on()`, `bg_off()`, `bg_on()`, `color_off()`, `color_pair()`, `fg_off()`, `fg_on()`, `make_bg()`, `make_fg()`, `make_style()`, `reset()`

**Examples**
```r
x <- style("Hello World!\n", fg="blue", bg=c(192,192,192), attr=c("ul", "st"))
cat(paste(x, "It is nice to meet you!"))
```
### Sym

**Unicode Symbols**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A named list containing the unicode character for various box drawing, mathematical, currency, astrological, and other symbols.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sym</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>A named list of characters</td>
</tr>
</tbody>
</table>

### term_dim

**Determine Terminal Size**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function determines the size of the terminal in number of rows and columns. The value may not be accurate in RStudio or the R GUI.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>term_dim()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>numeric vector (# of rows, # of columns)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>term_dim()</td>
</tr>
</tbody>
</table>
vline  

Vertical Line

Description
Vertical Line

Usage
vline(n, text = "|")

Arguments

- n  integer describing the character length of the line
- text  character to be repeated

Value
character string of length n, separated by "\n"

See Also
Other drawing functions: box_at(), draw_arc(), draw_bezier(), draw_circle(), draw_ellipse(),
draw_fn(), draw_lerp(), draw_path(), draw_ray(), draw_rect(), draw_shape(), fill_circle(),
fill_ellipse(), fill_rect(), fill_shape(), grid_at(), grid_mat(), hline_at(), hline(),
vline_at()

Examples
vline(4, "*")  # "*\n*\n*\n*"

vline_at  

Draw Vertical Line

Description
Draws a vertical line of length n at (row, col)

Usage
vline_at(yx, n, text = "|", ...)
Arguments

- **yx**: (row, col) coordinates where top of the line should be drawn.
- **n**: integer describing the character length of the line
- **text**: character to be repeated
- **...**: parameters that are passed to `style()`, including the foreground color `fg`, background color `bg`, and attribute `attr`

Value

- **NULL**

See Also

Other drawing functions: `box_at()`, `draw_arc()`, `draw bezier()`, `draw_circle()`, `draw ellipse()`, `draw fn()`, `draw lerp()`, `draw path()`, `draw ray()`, `draw rect()`, `draw shape()`, `fill circle()`, `fill ellipse()`, `fill rect()`, `fill shape()`, `grid at()`, `grid mat()`, `hline at()`, `hline()`, `vline()`

Examples

- `vline at(c(3,4),6,"|")` # print "|" at (3,4), ..., (8,4)

---

**wr**

### Write String to Terminal

**Description**

Writes a string of characters to the terminal at the current cursor position. `wr` accepts text colors and attributes, but these are reset to default afterwards if used.

**Usage**

```r
wr(text, fg = NA, bg = NA, attr = NA)
```

**Arguments**

- **text**: string to be printed to the Console
- **fg**: foreground color. See `fg_on` for more details.
- **bg**: background color. See `bg_on` for more details.
- **attr**: character attribute. See `attr_on` for more details.

**Value**

- **NULL**
See Also

Other writing functions: \texttt{wrat()}, \texttt{wrch()}, \texttt{wrkpl()}, \texttt{wrkp()}

Examples

\begin{verbatim}
    mv_to(5,4)
    wrch("h")
    wrch("e", fg="red")
    wr("llo World")
\end{verbatim}

\begin{verbatim}
wrapup
\end{verbatim}

\textit{Return Screen to Blank State}

Description

Function to be used at the end of a terminal function. It resets the colors and attributes to their default values, clears the screen, and reveals the cursor.

Usage

\begin{verbatim}
wrapup()
\end{verbatim}

Value

\begin{verbatim}
NULL
\end{verbatim}

wrat

\textit{Write At a Specific Location}

Description

Move cursor to specified location in the terminal screen, then print the supplied text. This function will only work in terminal, not the RStudio Console or R GUI.

Usage

\begin{verbatim}
wrat(yx, text, ...)
\end{verbatim}

Arguments

\begin{verbatim}
yx           numeric vector specifying the (row, col) coordinates to print at
text         text to be written at yx
...           colors and attributes added to text. See \texttt{wr}, \texttt{fg_on}, \texttt{bg_on}, and \texttt{attr_on} for more details.
\end{verbatim}
Details
The coordinates are given in matrix notation: \((row, column)\), with the top-left corner of the screen being \((1,1)\).

Value

NULL

See Also

Other writing functions: \texttt{wrch()}, \texttt{wrkpl()}, \texttt{wrkp()}, \texttt{wr()}

Examples

\begin{verbatim}
wrat(c(10,6), "CURSR")
wrat(c(4,1), "Hello World!", fg="red", attr=c("bf", "ul"))

mat <- rbind(c(5,2), c(10,5), c(1,19))
wrat(mat, "HI", fg="yellow")
\end{verbatim}

\texttt{wrch}

\texttt{Write Character to Terminal}

Description

Writes a single character to the terminal at the current cursor position. \texttt{wr} accepts text colors and attributes, but these are reset to default afterwards if used.

Usage

\texttt{wrch(chr, fg = NA, bg = NA, attr = NA)}

Arguments

\begin{itemize}
  \item \texttt{chr} character to be printed to the Console
  \item \texttt{fg} foreground color. See \texttt{fg_on} for more details.
  \item \texttt{bg} background color. See \texttt{bg_on} for more details.
  \item \texttt{attr} character attribute. See \texttt{attr_on} for more details.
\end{itemize}

Value

NULL

See Also

Other writing functions: \texttt{wrat()}, \texttt{wrkpl()}, \texttt{wrkp()}, \texttt{wr()}
Examples

```r
mv_to(5, 4)
wrch("h")
wrch("e", fg="red")
wr("llo World")
```
**Description**

Detect keypress and print it to the terminal screen, while invisibly returning the keypress. The user can specify which characters to ignore, and can also map keys to a list of functions. Any keypress mapped to a function will not be echoed to the screen.

**Usage**

\[
\text{wrkp}(\text{ignore} = "\text{escape}", \text{fn} = \text{list}, \ldots) 
\]

**Arguments**

- `ignore` vector of keypresses to ignore.
- `fn` list of functions, named by key, to be called when key is pressed.
- `...` parameters that are passed to `style()`, including the foreground color `fg`, background color `bg`, and attribute `attr`.

**Value**

NULL

**See Also**

Other writing functions: `wrat()`, `wrch()`, `wrkpl()`, `wr()`

**Examples**

```r
## Not run:
wkp(  
ignore="escape",  
fn = list(  
  enter = function(){mv_row(1)},  
  left = function(){mv(0, -1)},  
  right = function(){mv(0, 1)},  
  up = function(){mv(-1,0)},  
  down = function(){mv(1,0)},  
  space = function(){cat(" ")}),  
)  
## End(Not run)
```
Description

Detect keypress and print it to the terminal screen, while invisibly returning the keypress. The user can specify which characters to ignore, and can also map keys to a list of functions. Any keypress mapped to a function will not be echoed to the screen.

Usage

\texttt{wrkpl(escape = c("escape"), ignore = NA_character_, fn = list(), ...)}

Arguments

- \texttt{escape} vector of keypresses to escape the loop.
- \texttt{ignore} vector of keypresses to ignore.
- \texttt{fn} list of functions, named by key, to be called when key is pressed.
- \texttt{...} parameters that are passed to \texttt{style()}, including the foreground color \texttt{fg}, background color \texttt{bg}, and attribute \texttt{attr}

Value

\texttt{NULL}

See Also

Other writing functions: \texttt{wrat()}, \texttt{wrch()}, \texttt{wrkp()}, \texttt{wr()}

Examples

```R
## Not run:
wrkpl(
  escape = "escape",
  fn = list(
    enter = function(){mv_row(1)},
    left = function(){mv(0, -1)},
    right = function(){mv(0, 1)},
    up = function(){mv(-1,0)},
    down = function(){mv(1,0)},
    space = function(){cat(" ")}
  )
)

## End(Not run)
```
Index

* datasets
  Sym, 47

* drawing functions
  box_at, 6
  draw_arc, 9
  draw_bezier, 10
  draw_circle, 11
  draw_ellipse, 12
  draw_fn, 13
  draw_lerp, 14
  draw_path, 14
  draw_ray, 15
  draw_rect, 16
  draw_shape, 17
  fill_circle, 20
  fill_ellipse, 21
  fill_rect, 22
  fill_shape, 23
  grid_at, 25
  grid_mat, 26
  hline, 28
  hline_at, 28
  vline, 48
  vline_at, 48

* moving functions
  mv, 32
  mv_col, 33
  mv_row, 34
  mv_to, 35

* path-fitting functions
  path_arc, 36
  path_bezier, 36
  path_circle, 37
  path_ellipse, 38
  path_fill, 39
  path_fn, 39
  path_intersection, 40
  path_lerp, 41
  path_ray, 41

  path_rect, 42
  path_shape, 43

* style functions
  attr_off, 3
  attr_on, 4
  bg_off, 5
  bg_on, 5
  color_off, 8
  color_pair, 9
  fg_off, 19
  fg_on, 19
  make_bg, 30
  make_fg, 31
  make_style, 32
  reset, 44
  style, 46

* writing functions
  wr, 49
  wrat, 50
  wrch, 51
  wrkp, 53
  wrkpl, 54

  attr_off, 3, 4–6, 8, 9, 19, 20, 30–32, 44, 46
  attr_on, 3, 4, 5–9, 19, 20, 30–32, 44, 46,
    49–51
  bg_off, 3, 4, 5, 6, 8, 9, 19, 20, 30–32, 44, 46
  bg_on, 3–5, 5, 7–9, 19, 20, 30–32, 44, 46,
    49–51
  box_at, 6, 10–17, 21–23, 26–29, 48, 49
  clear, 7
  color_off, 3–6, 8, 9, 19, 20, 30–32, 44, 46
  color_pair, 3–6, 8, 9, 19, 20, 30–32, 44, 46
  draw_arc, 7, 9, 11–17, 21–23, 26–29, 48, 49
  draw_bezier, 7, 10, 11, 11–17, 21–23, 26–29, 48, 49
  draw_circle, 7, 10, 11, 11, 12–17, 21–23,
    26–29, 48, 49
draw_ellipse, 7, 10, 11, 12, 13–17, 21–23, 26–29, 48, 49
draw_fn, 7, 10–12, 13, 14–17, 21–23, 26–29, 48, 49
draw_lerp, 7, 10–13, 14, 15–17, 21–23, 26–29, 48, 49
draw_path, 7, 10–14, 14, 16, 17, 21–23, 26–29, 48, 49
draw_ray, 7, 10–15, 15, 16, 17, 21–23, 26–29, 48, 49
draw_rect, 7, 10–16, 16, 17, 21–23, 26–29, 48, 49
draw_shape, 7, 10–16, 17, 21–23, 26–29, 48, 49
erase, 18
example_luckynumber, 18
fg_off, 3–6, 8, 9, 19, 20, 30–32, 44, 46
fg_on, 3–9, 19, 19, 30–32, 44, 46, 49–51
fill_circle, 7, 10–17, 20, 21–23, 26–29, 48, 49
fill_ellipse, 7, 10–17, 21, 22, 23, 26–29, 48, 49
fill_rect, 7, 10–17, 21, 22, 23, 26–29, 48, 49
fill_shape, 7, 10–17, 21, 22, 23, 26–29, 48, 49
getkp, 24
gtkpl, 24
grid_at, 7, 10–17, 21–23, 25, 27–29, 48, 49
grid_mat, 7, 10–17, 21–23, 26, 28, 29, 48, 49
hide_cursor, 27, 45
hline, 7, 10–17, 21–23, 26, 28, 29, 48, 49
hline_at, 7, 10–17, 21–23, 26–28, 28, 48, 49
in.term, 29
load_cursor, 30, 45
make_bg, 3–6, 8, 9, 19, 20, 30, 31, 32, 44, 46
make_fg, 3–6, 8, 9, 19, 20, 30, 31, 32, 44, 46
make_style, 3–6, 8, 9, 19, 20, 30, 31, 32, 44, 46
mv, 32, 34, 35
mv_col, 33, 34, 35
mv_row, 33, 34, 35
mv_to, 33, 34, 35
path_arc, 36, 37–43
path_bezier, 36, 37–43
path_circle, 36, 37, 38–43
path_ellipse, 36, 37, 38, 39–43
path_fill, 36–38, 39, 40–43
path_fn, 36–39, 39, 40–43
path_intersection, 36–40, 40, 41–43
path_lerp, 36–40, 41, 42, 43
path_ray, 36–41, 41, 42, 43
path_rect, 36–42, 42, 43
path_shape, 36–42, 43
repch, 44
reset, 3–6, 8, 9, 19, 20, 30–32, 44, 46
save_cursor, 30, 45
show_cursor, 27, 45
style, 3–6, 8, 9, 19, 20, 30–32, 44, 46
Sym, 47
term_dim, 47
vline, 7, 10–17, 21–23, 26–29, 48, 49
vline_at, 7, 10–17, 21–23, 26–29, 48, 49
wr, 49, 50, 51, 53, 54
wrapup, 50
wrchat, 50, 51, 53, 54
wrch, 50, 51, 53, 54
wrkpl, 50, 51, 53, 54
wrkp, 50, 51, 53, 54
wrapup, 50
wrch, 50, 51, 53, 54
wrkpl, 50, 51, 53, 54
wrkp, 50, 51, 53, 54