SB-SCREEN Documentation

Brian Mastenbrook

Table of Contents

1 Initializing the screen 1

2 Positioning the cursor 2

3 Drawing text 3

4 Color 4

5 Window sizing 5

6 Key handling 6

Appendix A Function Index 7

1 Initializing the screen

To initialize the screen, you must first have a screen object on hand. For the TTY backend, this type is TTY-SCREEN. Other backends may exist in the future. Make a screen object with `make-instance`, and then use `initialize-screen` to initialize the screen and allow drawing. `release-screen` is an implementation-defined function which should be called to release whatever resources are held once the screen session is finished.

sb-screen:initialize-screen *screen* &key (*abort-char-code* **-1**) [Function]

Lambda-list: screen &key &allow-other-keys

Initialize the screen for the given screen object. Extra keyword arguments are defined by the given screen object type.

sb-screen:release-screen *screen* [Function]

Lambda-list: screen

Release the given screen object. What this means is defined by the type of the supplied screen object.

2 Positioning the cursor

The cursor is the main point where the user sees the insertion cursor and where all text drawing operations originate.

sb-screen:set-cursor *screen row col* [Function]

Lambda-list: screen row col

Move the user's insertion cursor to the given row and column.

sb-screen:get-cursor *screen* [Function]

Lambda-list: screen

Return, in multiple values, the row and column that the user's insertion point is at.

3 Drawing text

These functions are the main API for drawing text on the screen with SB-SCREEN, clearing portions of text, and drawing special glyphs.

sb-screen:clear-screen *screen* [Function]

Lambda-list: screen

Clear the given screen.

sb-screen:finish-screen *screen* [Function]

Lambda-list: screen

Finish all output operations to the given screen.

sb-screen:write-string-at-cursor *screen string* [Function]

Lambda-list: screen string

Write the given string to the screen at the cursor position. If the input includes a newline character or exceeds the number of columns following the cursor position, the output will be truncated, not wrapped.

sb-screen:erase-from-cursor-to-eol *screen* [Function]

Lambda-list: screen

Erase the screen between the cursor point and the end of the row.

sb-screen:erase-from-cursor-to-eos *screen* [Function]

Lambda-list: screen

Erase the screen between the cursor point and the end of the screen.

sb-screen:draw-line-at-cursor *screen direction length* [Function]

Lambda-list: screen direction length

If possible, draw a line of the given length and running in the supplied direction on the screen. Valid directions are :horizontal and :vertical.

4 Color

These functions allow the use of color in on-screen drawing, and the determination of which colors are valid for text.

sb-screen:valid-color-p *screen sym type* [Function]

Lambda-list: screen name type

Returns true iff the color designated by the symbol name names a valid color of the specified type. Valid values for type are :foreground and :background.

sb-screen:valid-colors *screen type* [Function]

Lambda-list: screen type

Returns a possibly non-exhaustive list of colors which are valid for the given type on the given screen. Valid types are :foreground and :background.

sb-screen:set-color *screen foreground background* [Function]

Lambda-list: screen foreground background

Set the color of the following text drawing operations to the given colors named by the symbols foreground and background.

sb-screen:set-to-default-color *screen* [Function]

Lambda-list: screen

Set the color of the following text drawing operations to the system default foreground and background.

5 Window sizing

These functions allow the program to determine how large the window is, and recieve callbacks when the window size is changed.

sb-screen:get-screen-size *screen* [Function]

Lambda-list: screen

Return multiple values containing the number of rows and columns of the screen's current size.

sb-screen:window-resize-hook *screen* [Function]

Lambda-list: screen

Return the current hook which is invoked when the window is resized. Use (setf window-resize-hook) to set the hook or remove it (by setting it to nil).

6 Key handling

These functions allow the program to manipulate the representation of key events and recieve callbacks when keyboard events occur.

sb-screen:encode-key *screen key* [Function]

Lambda-list: screen key

Encode the given representation of a key into a numeric value.

sb-screen:decode-key *screen keysym* [Function]

Lambda-list: screen key

Decode the given number into a representation of it as a key.

sb-screen:key-hook *screen* [Function]

Lambda-list: screen

Returns the current hook that is invoked when a key is pressed. Use (setf key-hook) to set the hook or clear it (by setting it to nil).

Appendix A Function Index

C

clear-screen 3

D

decode-key 6
draw-line-at-cursor 3

E

encode-key 6
erase-from-cursor-to-eol 3
erase-from-cursor-to-eos 3

F

finish-screen 3

G

get-cursor 2
get-screen-size 5

I

initialize-screen 1

K

key-hook 6

R

release-screen 1

S

sb-screen:clear-screen 3
sb-screen:decode-key 6
sb-screen:draw-line-at-cursor 3
sb-screen:encode-key 6
sb-screen:erase-from-cursor-to-eol 3
sb-screen:erase-from-cursor-to-eos 3
sb-screen:finish-screen 3
sb-screen:get-cursor 2
sb-screen:get-screen-size 5
sb-screen:initialize-screen 1
sb-screen:key-hook 6
sb-screen:release-screen 1
sb-screen:set-color 4
sb-screen:set-cursor 2
sb-screen:set-to-default-color 4
sb-screen:valid-color-p 4
sb-screen:valid-colors 4
sb-screen:window-resize-hook 5
sb-screen:write-string-at-cursor 3
set-color 4
set-cursor 2
set-to-default-color 4

V

valid-color-p 4
valid-colors 4

W

window-resize-hook 5
write-string-at-cursor 3