NAME

cdk_buttonbox − Creates a managed curses buttonbox widget.

SYNOPSIS

cc [*flag* . . .] *file* . . . **−lcdk** [*library* . . .]

#include <cdk.h>

int activateCDKButtonbox (
CDKBUTTONBOX ****buttonbox*,**
chtype * *actions***);**

void destroyCDKButtonbox (
CDKBUTTONBOX ****buttonbox*);**

void drawCDKButtonbox (
CDKBUTTONBOX ****buttonbox*,**
boolean *box***);**

void drawCDKButtonboxButtons (
CDKBUTTONBOX ****buttonbox*);**

void eraseCDKButtonbox (
CDKBUTTONBOX ****buttonbox*);**

boolean getCDKButtonboxBox (
CDKBUTTONBOX ****buttonbox*);**

int getCDKButtonboxButtonCount (
CDKBUTTONBOX ****buttonbox*);**

int getCDKButtonboxCurrentButton (
CDKBUTTONBOX ****buttonbox*);**

chtype getCDKButtonboxHighlight (
CDKBUTTONBOX ****buttonbox*);**

int injectCDKButtonbox (
CDKBUTTONBOX ****buttonbox*,**
chtype *input***);**

void moveCDKButtonbox (
CDKBUTTONBOX ****buttonbox*,**
int *xpos***,**
int *ypos***,**
boolean *relative***,**
boolean *refresh***);**

CDKBUTTONBOX *newCDKButtonbox (
CDKSCREEN ****cdkscreen*,**
int *xpos***,**
int *ypos***,**
int *height***,**
int *width***,**
char * *title***,**
int *rows***,**
int *cols***,**
char ** *buttons***,**
int *buttonCount***,**
chtype *highlight***,**
boolean *box***,**
boolean *shadow***);**

void positionCDKButtonbox (
CDKBUTTONBOX ****buttonbox*);**

void setCDKButtonbox (
CDKBUTTONBOX ****buttonbox*,**
chtype *highlight***,**
boolean *box***);**

void setCDKButtonboxBackgroundAttrib (
CDKBUTTONBOX ****buttonbox*,**
chtype *attribute***);**

void setCDKButtonboxBackgroundColor (
CDKBUTTONBOX ****buttonbox*,**
char * *color***);**

void setCDKButtonboxBox (
CDKBUTTONBOX ****buttonbox*,**
boolean *box***);**

void setCDKButtonboxBoxAttribute (
CDKBUTTONBOX ****buttonbox*,**
chtype *character***);**

void setCDKButtonboxCurrentButton (
CDKBUTTONBOX ****buttonbox*,**
int *button***);**

void setCDKButtonboxHighlight (
CDKBUTTONBOX ****buttonbox*,**
chtype *highlight***);**

void setCDKButtonboxHorizontalChar (
CDKBUTTONBOX ****buttonbox*,**
chtype *character***);**

void setCDKButtonboxLLChar (
CDKBUTTONBOX ****buttonbox*,**
chtype *character***);**

void setCDKButtonboxLRChar (
CDKBUTTONBOX ****buttonbox*,**
chtype *character***);**

void setCDKButtonboxPostProcess (
CDKBUTTONBOX ****buttonbox*,**
PROCESSFN *callback***,**
void * *data***);**

void setCDKButtonboxPreProcess (
CDKBUTTONBOX ****buttonbox*,**
PROCESSFN *callback***,**
void * *data***);**

void setCDKButtonboxULChar (
CDKBUTTONBOX ****buttonbox*,**
chtype *character***);**

void setCDKButtonboxURChar (
CDKBUTTONBOX ****buttonbox*,**
chtype *character***);**

void setCDKButtonboxVerticalChar (
CDKBUTTONBOX ****buttonbox*,**
chtype *character***);**

DESCRIPTION

The Cdk buttonbox widget creates a buttonbox box with a message and a varied number of buttons to choose from. The following functions create or manipulate the Cdk buttonbox box widget.

AVAILABLE FUNCTIONS

activateCDKButtonbox

activates the buttonbox widget and lets the user interact with the widget. The parameter **buttonbox** is a pointer to a non-NULL buttonbox widget. If the **actions** parameter is passed with a non-NULL value, the characters in the array will be injected into the widget. To activate the widget interactively pass in a *NULL* pointer for **actions**. If the character entered into this widget is *RETURN* or *TAB* then this function will return a value from 0 to the number of buttons -1, representing the button selected. It will also set the widget data *exitType* to *vNORMAL*. If the character entered into this widget was *ESCAPE* then the widget will return a value of -1 and the widget data *exitType* will be set to *vESCAPE_HIT*.

destroyCDKButtonbox

removes the widget from the screen and releases any memory the object used.

drawCDKButtonbox

draws the buttonbox widget on the screen. If the **box** parameter is true, the widget is drawn with a box.

drawCDKButtonboxButtons

draws the buttons.

eraseCDKButtonbox

removes the widget from the screen. This does *NOT* destroy the widget.

getCDKButtonboxBox

returns true if the widget will be drawn with a box around it.

getCDKButtonboxButtonCount

returns the number of buttons in the button box.

getCDKButtonboxCurrentButton

returns the current button-number.

getCDKButtonboxHighlight

returns the highlight attribute of the widget.

injectCDKButtonbox

injects a single character into the widget. The parameter **buttonbox** is a pointer to a non-NULL buttonbox widget. The parameter **character** is the character to inject into the widget. The return value and side-effect (setting the widget data *exitType*) depend upon the injected character:

RETURN or *TAB*

the function returns a value ranging from zero to one less than the number of buttons, representing the button selected. The widget data *exitType* is set to *vNORMAL*.

ESCAPE

the function returns -1. *vESCAPE_HIT*. The widget data *exitType* is set to *vESCAPE_HIT*.

Otherwise

unless modified by preprocessing, postprocessing or key bindings, the function returns -1. The widget data *exitType* is set to *vEARLY_EXIT*.

moveCDKButtonbox

moves the given widget to the given position. The parameters **xpos** and **ypos** are the new position of the widget. The parameter **xpos** may be an integer or one of the pre-defined values *TOP*, *BOTTOM*, and *CENTER*. The parameter **ypos** may be an integer or one of the pre-defined values *LEFT*, *RIGHT*, and *CENTER*. The parameter **relative** states whether the **xpos**/**ypos** pair is a relative move or an absolute move. For example, if **xpos** = 1 and **ypos** = 2 and **relative** = **TRUE**, then the widget would move one row down and two columns right. If the value of **relative** was **FALSE** then the widget would move to the position (1,2). Do not use the values *TOP*, *BOTTOM*, *LEFT*, *RIGHT*, or *CENTER* when **relative** = *TRUE*. (weird things may happen). The final parameter **refresh** is a boolean value which states whether the widget will get refreshed after the move.

newCDKButtonbox
creates a pointer to a buttonbox widget. Parameters:

screen
is the screen you wish this widget to be placed in.

xpos controls the placement of the object along the horizontal axis. It may be an integer or one of the pre-defined values *LEFT*, *RIGHT*, and *CENTER*.

ypos controls the placement of the object along the vertical axis. It be an integer value or one of the pre-defined values *TOP*, *BOTTOM*, and *CENTER*.

height and

width
control the height and width of the widget.

title is the title of the widget.

rows
is the number of rows of buttons.

cols sets the number of columns.

buttons
is an array containing the button labels.

buttonCount
is the number of elements in the **buttons** array.

highlight
is the attribute of the currently highlighted button.

box is true if the widget should be drawn with a box around it.

shadow
turns the shadow on or off around this widget.

If the widget could not be created then a *NULL* pointer is returned.

positionCDKButtonbox
allows the user to move the widget around the screen via the cursor/keypad keys. See **cdk_position (3)** for key bindings.

setCDKButtonbox
lets the programmer modify certain elements of an existing buttonbox widget. The parameter names correspond to the same parameter names listed in the **newCDKButtonbox** function.

setCDKButtonboxBackgroundAttrib
sets the background attribute of the widget. The parameter **attribute** is a curses attribute, e.g., A_BOLD.

setCDKButtonboxBackgroundColor
sets the background color of the widget. The parameter **color** is in the format of the Cdk format strings. (See *cdk_display*).

setCDKButtonboxBox
sets true if the widget will be drawn with a box around it.

setCDKButtonboxBoxAttribute
sets the attribute of the box.

setCDKButtonboxCurrentButton
sets the current button-number for the box.

setCDKButtonboxHighlight
sets the highlight attribute of the selected button.

setCDKButtonboxHorizontalChar
sets the horizontal drawing character for the box to the given character.

setCDKButtonboxLLChar
sets the lower left hand corner of the widget's box to the given character.

setCDKButtonboxLRChar
sets the lower right hand corner of the widget's box to the given character.

setCDKButtonboxPostProcess
allows the user to have the widget call a function after the key has been applied to the widget. The parameter **function** is the callback function. (See *cdk_process*).

setCDKButtonboxPreProcess
allows the user to have the widget call a function after a key is hit and before the key is applied to the widget. The parameter **function** is the callback function. (See *cdk_process*).

setCDKButtonboxULChar
sets the upper left hand corner of the widget's box to the given character.

setCDKButtonboxURChar
sets the upper right hand corner of the widget's box to the given character.

setCDKButtonboxVerticalChar
sets the vertical drawing character for the box to the given character.

KEY BINDINGS

When the widget is activated there are several default key bindings which will help the user enter or manipulate the information quickly. The following table outlines the keys and their actions for this widget.

Key	**Action**
Left Arrow	Selects the button to the left of the current button.
Right Arrow	Selects the button to the right of the current button.
Tab	Selects the button to the right of the current button.
Space	Selects the button to the right of the current button.
Return	Exits the widget and returns the index of the selected button. This also sets the widget data *exitType* to *vNORMAL*.
Tab	Exits the widget and returns the index of the selected button. This also sets the widget data *exitType* to *vNORMAL*.
Escape	Exits the widget and returns -1. This also sets the widget data *exitType* to *vESCAPE_HIT*.
Ctrl-L	Refreshes the screen.

SEE ALSO

cdk(3), **cdk_binding**(3), **cdk_display**(3), **cdk_position**(3), **cdk_process**(3), **cdk_screen**(3)