NAME

cdk_selection − curses selection list widget.

SYNOPSIS

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

#include <cdk.h>

int activateCDKSelection (
CDKSELECTION ****selection*,**
chtype * *actions***);**

void destroyCDKSelection (
CDKSELECTION ****selection*);**

void drawCDKSelection (
CDKSELECTION ****selection*,**
boolean *box***);**

void eraseCDKSelection (
CDKSELECTION ****selection*);**

boolean getCDKSelectionBox (
CDKSELECTION ****selection*);**

int getCDKSelectionChoice (
CDKSELECTION ****selection*,**
int *index***);**

int *getCDKSelectionChoices (
CDKSELECTION ****selection*);**

int getCDKSelectionCurrent (
CDKSELECTION ****selection*);**

chtype getCDKSelectionHighlight (
CDKSELECTION ****selection*);**

int getCDKSelectionItems (
CDKSELECTION ****selection*,**
char *****list*);**

int getCDKSelectionMode (
CDKSELECTION ****selection*,**
int *index***);**

int *getCDKSelectionModes (
CDKSELECTION ****selection*);**

char *getCDKSelectionTitle (
CDKSELECTION ****selection*);**

int injectCDKSelection (
CDKSELECTION ****selection*,**
chtype *input***);**

void moveCDKSelection (
CDKSELECTION ****selection*,**
int *xpos***,**
int *ypos***,**
boolean *relative***,**
boolean *refresh***);**

CDKSELECTION *newCDKSelection (
CDKSCREEN ****cdkscreen*,**
int *xpos***,**
int *ypos***,**
int *spos***,**
int *height***,**

int *width*,
**char ***_title_,
**char ****_selectionList_,
int *selectionListLength*,
**char ****_choiceList_,
int *choiceListLength*,
chtype *highlight*,
boolean *box*,
boolean *shadow*);

void positionCDKSelection (
**CDKSELECTION ***_selection_);

void setCDKSelection (
**CDKSELECTION ***_selection_,
chtype *highlight*,
**int ***_defChoices_,
boolean *box*);

void setCDKSelectionBackgroundAttrib (
**CDKSELECTION ***_selection_,
chtype *attribute*);

void setCDKSelectionBackgroundColor (
**CDKSELECTION ***_selection_,
char * *color*);

void setCDKSelectionBox (
**CDKSELECTION ***_selection_,
boolean *boxWidget*);

void setCDKSelectionBoxAttribute (
**CDKSELECTION ***_selection_,
chtype *character*);

void setCDKSelectionChoice (
**CDKSELECTION ***_selection_,
int *index*,
int *choice*);

void setCDKSelectionChoices (
**CDKSELECTION ***_selection_,
**int ***_choices_);

void setCDKSelectionCurrent (
**CDKSELECTION ***_selection_,
int*index*);

void setCDKSelectionHighlight (
**CDKSELECTION ***_selection_,
chtype *highlight*);

void setCDKSelectionHorizontalChar (
**CDKSELECTION ***_selection_,
chtype *character*);

void setCDKSelectionItems (
**CDKSELECTION ***_selection_,
**char ****_list_,
int *listSize*);

void setCDKSelectionLLChar (
**CDKSELECTION ***_selection_,
chtype *character*);

void setCDKSelectionLRChar (
CDKSELECTION ****selection*,**
chtype *character***);**

void setCDKSelectionMode (
CDKSELECTION ****selection*,**
int *index***,**
int *mode***);**

void setCDKSelectionModes (
CDKSELECTION ****selection*,**
int ****modes*);**

void setCDKSelectionPostProcess (
CDKSELECTION ****selection*,**
PROCESSFN *callback***,**
void * *data***);**

void setCDKSelectionPreProcess (
CDKSELECTION ****selection*,**
PROCESSFN *callback***,**
void * *data***);**

void setCDKSelectionTitle (
CDKSELECTION ****selection*,**
char ****title*);**

void setCDKSelectionULChar (
CDKSELECTION ****selection*,**
chtype *character***);**

void setCDKSelectionURChar (
CDKSELECTION ****selection*,**
chtype *character***);**

void setCDKSelectionVerticalChar (
CDKSELECTION ****selection*,**
chtype *character***);**

DESCRIPTION

The Cdk selection widget creates a selection list. The following functions create or manipulate the Cdk selection list widget.

AVAILABLE FUNCTIONS

activateCDKSelection

activates the selection widget and lets the user interact with the widget. The parameter **selection** is a pointer to a non-NULL selection 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 1. Itm 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*.

destroyCDKSelection

removes the widget from the screen and frees memory the object used.

drawCDKSelection

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

eraseCDKSelection

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

getCDKSelectionBox

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

getCDKSelectionChoice
returns the selection choice at the given index.

getCDKSelectionChoices
returns an array of the current selection choices for the widget.

getCDKSelectionCurrent
returns the current selection index.

getCDKSelectionHighlight
returns the attribute of the highlight bar.

getCDKSelectionItems
copies the selection-list items into the caller's array and returns the number of items in the list.

getCDKSelectionMode
returns the selection mode at the given index.

getCDKSelectionModes
returns an array of the current modes for the widget.

getCDKSelectionTitle
returns the first line of the title of the selection widget. The caller must free the returned value.

injectCDKSelection
injects a single character into the widget. The parameter **selection** is a pointer to a non-NULL selection 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 1. 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*.

moveCDKSelection
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.

newCDKSelection
creates a selection widget and returns a pointer to it. 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 may be an integer or one of the pre-defined values *TOP*, *BOTTOM*, and *CENTER*.

spos is where the scroll bar is to be placed. It may be one of three values:

LEFT
puts the scroll bar on the left of the scrolling list. *RIGHT* puts the scroll bar on the right side of the list, and

NONE
does not add a scroll bar.

height and

width
control the height and width of the widget. If you provide a value of zero for either of the height or the width, the widget will be created with the full width and height of the screen. If you provide a negative value, the widget will be created the full height or width minus the value provided.

title is the string which to display at the top of the widget. The title can be more than one line; just provide a carriage return character at the line break.

selectionList
is the list of items to display in the selection list

selectionListLength
is the number of elements in the given list.

choiceList
is the list of choices that will be selected when the user presses the space bar.

choiceListLength
is the length of this list.

highlight
specifies the display attribute of the currently selected item.

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.

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

setCDKSelection
lets the programmer modify certain elements of an existing selection widget. The parameter names correspond to the same parameter names listed in the **newCDKSelection** function.

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

setCDKSelectionBackgroundColor
sets the background color of the widget. The parameter **color** is in the format of the Cdk format strings. See **cdk_display (3)**.

setCDKSelectionBox
sets whether the widget will be drawn with a box around it.

setCDKSelectionBoxAttribute
sets the attribute of the box.

setCDKSelectionChoice
sets the selection choice value at the given index.

setCDKSelectionChoices
sets the selection choice values of the widget.

setCDKSelectionCurrent
sets the current selection index.

setCDKSelectionHighlight
sets the attribute of the highlight bar.

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

setCDKSelectionItems
sets the selection list items.

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

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

setCDKSelectionMode
sets the selection mode at the given index.

setCDKSelectionModes
sets the selection mode of the elements of the widget. There are two acceptable values for the modes: 0 which allows the user to change the selection value at the given index; and 1 which sets the element to a read-only state.

setCDKSelectionPostProcess
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. The parameter **data** points to data passed to the callback function. To learn more about post-processing see *cdk_process (3)*.

setCDKSelectionPreProcess
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. The parameter **data** is a pointer to *void*. To learn more about pre-processing see *cdk_process (3)*.

setCDKSelectionTitle
set the selection list's title.

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

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

setCDKSelectionVerticalChar
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 help the user enter or manipulate the information quickly:

Key	Action
Left Arrow	Shift the whole list left one column.
Right Arrow	Shift the whole list right one column.
Up Arrow	Select the previous item in the list.
Down Arrow	Select the next item in the list.
Prev Page Ctrl-B	Scroll one page backward.
Next Page Ctrl-F	Scroll one page forward.
1 < g Home	Move to the first element in the list.
> G End	Move to the last element in the list.
$	Shift the whole list to the far right.
|	Shift the whole list to the far left.
Space	Cycles to the next choice on the current item.
Return	Exit the widget and return 1. Also set the widget data *exitType* to *vNORMAL*.
Tab	Exit the widget and return 1. Also set the widget data *exitType* to *vNORMAL*.
Escape	Exit the widget and return -1. Also set 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_screen**(3)