NAME

cdk_scroll − curses scrolling list widget.

SYNOPSIS

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

#include <cdk.h>

int activateCDKScroll (
CDKSCROLL ****scroll*,**
chtype * *actions***);**

void addCDKScrollItem (
CDKSCROLL ****scroll*,**
char ****item*);**

void deleteCDKScrollItem (
CDKSCROLL ****scroll*,**
int *position***);**

void destroyCDKScroll (
CDKSCROLL ****scroll*);**

void drawCDKScroll (
CDKSCROLL ****scroll*,**
boolean *box***);**

void eraseCDKScroll (
CDKSCROLL ****scroll*);**

boolean getCDKScrollBox (
CDKSCROLL ****scroll*);**

int getCDKScrollCurrent(
CDKSCROLL ****scroll*);**

int getCDKScrollCurrentItem (
CDKSCROLL ****widget*);**

chtype getCDKScrollHighlight (
CDKSCROLL ****scroll*,**
chtype *highlight***);**

int getCDKScrollCurrentTop (
CDKSCROLL ****widget*);**

int getCDKScrollItems (
CDKSCROLL ****scroll*,**
char *****itemList*);**

int injectCDKScroll (
CDKSCROLL ****scroll*,**
chtype *input***);**

void insertCDKScrollItem (
CDKSCROLL ****scroll*,**
char ****item*);**

void moveCDKScroll (
CDKSCROLL ****scroll*,**
int *xpos***,**
int *ypos***,**
boolean *relative***,**
boolean *refresh***);**

CDKSCROLL *newCDKScroll (
CDKSCREEN ****cdkscreen*,**
int *xpos***,**
int *ypos***,**

int *spos*,
int *height*,
int *width*,
char * *title*,
char ** *itemList*,
int *items*,
boolean *numbers*,
chtype *highlight*,
boolean *box*,
boolean *shadow*);

void positionCDKScroll (
CDKSCROLL * *scroll*);

void setCDKScroll (
CDKSCROLL * *scroll*,
char ** *itemList*,
int *items*,
boolean *numbers*,
chtype *highlight*,
boolean *box*);

void setCDKScrollBackgroundAttrib (
CDKSCROLL * *scroll*,
chtype *attribute*);

void setCDKScrollBackgroundColor (
CDKSCROLL * *scroll*,
char * *color*);

void setCDKScrollBox (
CDKSCROLL * *scroll*,
boolean *box*);

void setCDKScrollBoxAttribute (
CDKSCROLL * *scroll*,
chtype *character*);

void setCDKScrollCurrent(
CDKSCROLL * *scroll*,
int*item*);

void setCDKScrollCurrentItem (
CDKSCROLL * *widget*,
int*item*);

int getCDKScrollCurrentTop (
CDKSCROLL * *widget*);

void setCDKScrollHighlight (
CDKSCROLL * *scroll*,
chtype *highlight*);

void setCDKScrollHorizontalChar (
CDKSCROLL * *scroll*,
chtype *character*);

void setCDKScrollItems (
CDKSCROLL * *scroll*,
char ** *itemList*,
int *listSize*,
boolean *numbers*);

void setCDKScrollLLChar (
CDKSCROLL * *scroll*,
chtype *character*);

void setCDKScrollLRChar (
CDKSCROLL ****scroll*,**
chtype *character***);**

void setCDKScrollPosition (
CDKSCROLL ****scroll*,**
int*item***);**

void setCDKScrollPostProcess (
CDKSCROLL ****scroll*,**
PROCESSFN *callback***,**
void * *data***);**

void setCDKScrollPreProcess (
CDKSCROLL ****scroll*,**
PROCESSFN *callback***,**
void * *data***);**

void setCDKScrollULChar (
CDKSCROLL ****scroll*,**
chtype *character***);**

void setCDKScrollURChar (
CDKSCROLL ****scroll*,**
chtype *character***);**

void setCDKScrollVerticalChar (
CDKSCROLL ****scroll*,**
chtype *character***);**

DESCRIPTION

The Cdk scroll widget creates a scrolling list. The following are functions which create or manipulate the Cdk scrolling list widget.

AVAILABLE FUNCTIONS

activateCDKScroll

activates the scroll widget and lets the user interact with the widget. The parameter **scroll** points to a non-NULL scroll 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 items-1, representing the item 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*.

addCDKScrollItem

allows the user to add an item into an existing scrolling list. The **scroll** parameter points to the scrolling list to add the item to. The parameter **item** is a *char ** representing the new item to add. The item is always added to the end of the list.

deleteCDKScrollItem

allows the user to add an item into an existing scrolling list. The **scroll** parameter points to the scrolling list to add the item to. The parameter **f2position** is an *int* which specifies which element to remove.

destroyCDKScroll

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

drawCDKScroll

draws the scroll widget on the screen. If the **box** option is true, the widget is drawn with a box.

eraseCDKScroll

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

getCDKScrollBox

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

getCDKScrollCurrent
returns the current item's index.

getCDKScrollCurrentItem
returns the current item number in the scroller.

getCDKScrollHighlight
returns the attribute of the highlight bar.

getCDKScrollCurrentTop
returns the top line of the scroller, counting from zero.

getCDKScrollItems
fills the parameter **itemList** with the contents of the scrolling list. It returns the number of elements in the scrolling list.

injectCDKScroll
injects a single character into the widget. The parameter **scroll** points to a non-NULL scroll 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 items, representing the item selected. The widget data *exitType* is set to *vNORMAL*.

ESCAPE
the function returns -1. 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*.

insertCDKScrollItem
allows the user to add an item into an existing scrolling list. The **scroll** parameter points to the scrolling list to add the item to. The parameter **item** is a *char* * representing the new item to add. The item is always added before the current item in the list.

moveCDKScroll
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.

newCDKScroll
creates a scroll widget and returns a pointer to it. Parameters:

screen
parameter is the screen you wish this widget to be placed in. The parameter **xpos** controls the placement of the object along the horizontal axis. This parameter 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. This parameter may be an integer value or one of the pre-defined values *TOP*, *BOTTOM*, and *CENTER*.

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

LEFT,
which puts the scroll bar on the left of the scrolling list.

RIGHT
which puts the scroll bar on the right side of the list, and

NONE which 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 will be displayed at the top of the widget. The title can be more than one line; just provide a carriage return character at the line break.

itemList
is the list of items to be displayed in the scrolling list.

items
is the number of elements in the given list.

numbers
is true if you want the items in the list to have a number attached to the front of the list items.

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
is true to turn the shadow on around this widget.

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

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

setCDKScroll
lets the programmer modify certain elements of an existing scroll widget. The parameter names correspond to the same parameter names listed in the **newCDKScroll** function.

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

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

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

setCDKScrollBoxAttribute
sets the attribute of the box.

setCDKScrollCurrent
sets the index for the current item.

setCDKScrollCurrentItem
set the current item number in the scroller.

setCDKScrollCurrentTop
set the top line number of the scroller.

setCDKScrollHighlight
sets the attribute of the highlight bar.

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

setCDKScrollItems
sets the contents of the scrolling list.

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

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

setCDKScrollPosition
sets the current item in the widget to the given position.

setCDKScrollPostProcess
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)*.

setCDKScrollPreProcess
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** points to data passed to the callback function. To learn more about pre-processing see *cdk_process (3)*.

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

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

setCDKScrollVerticalChar
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	Shift the list left one column.
Right Arrow	Shift the 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 list to the far right.
|	Shift the list to the far left.
Return	Exit the widget and return the index of the selected item. Also set the widget data *exitType* to *vNORMAL*.
Tab	Exit the widget and return the index of the selected item. 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)