NAME

cdk_swindow − a curses scrolling window widget.

SYNOPSIS

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

#include <cdk.h>

void activateCDKSwindow (
CDKSWINDOW ****swindow*,**
chtype * *actions***);**

void addCDKSwindow (
CDKSWINDOW ****swindow*,**
char ****info*,**
int *insertPosition***);**

void cleanCDKSwindow (
CDKSWINDOW ****swindow*);**

int execCDKSwindow (
CDKSWINDOW ****swindow*,**
char * *command***,**
int *insertPosition***);**

void destroyCDKSwindow (
CDKSWINDOW ****swindow*);**

void drawCDKSwindow (
CDKSWINDOW ****swindow*,**
boolean *box***);**

void dumpCDKSwindow (
CDKSWINDOW ****swindow*,**
char * *filename***);**

void eraseCDKSwindow (
CDKSWINDOW ****swindow*);**
void execCDKSwindow (
**CDKSWINDOW ****swindow,*
**char ****command,*
int *insertPosition);*
boolean getCDKSwindowBox (
**CDKSWINDOW ****swindow);*

chtype **getCDKSwindowContents (
CDKSWINDOW ****swindow*);**

int injectCDKSwindow (
CDKSWINDOW ****swindow*,**
chtype *input***);**

void jumpToLineCDKSwindow (
CDKSWINDOW ****swindow*,**
int *line***);**

void loadCDKSwindowInformation (
CDKSWINDOW ****swindow*);**

void moveCDKSwindow (
CDKSWINDOW ****swindow*,**
int *xpos***,**
int *ypos***,**
boolean *relative***,**
boolean *refresh***);**

CDKSWINDOW *newCDKSwindow (
CDKSCREEN ***cdkscreen*,**
int *xpos***,**
int *ypos***,**
int *height***,**
int *width***,**
char ***title*,**
int *saveLines***,**
boolean *box***,**
boolean *shadow***);**

void positionCDKSwindow (
CDKSWINDOW ***swindow*);**

void saveCDKSwindowInformation (
CDKSWINDOW ***swindow*);**

void setCDKSwindow (
CDKSWINDOW ***swindow*,**
char ****info*,*
int *lines***,**
boolean *box***);**

void setCDKSwindowBackgroundAttrib (
CDKSWINDOW ***swindow*,**
chtype *attribute***);**

void setCDKSwindowBackgroundColor (
CDKSWINDOW ***swindow*,**
char * *color***);**
void setCDKSwindowBox (
**CDKSWINDOW ***swindow,*
boolean *boxWidget);*

void setCDKSwindowBoxAttribute (
CDKSWINDOW ***swindow*,**
chtype *character***);**

void setCDKSwindowContents (
CDKSWINDOW ***swindow*,**
char ****info*,*
int *lines***);**

void setCDKSwindowHorizontalChar (
CDKSWINDOW ***swindow*,**
chtype *character***);**

void setCDKSwindowLLChar (
CDKSWINDOW ***swindow*,**
chtype *character***);**

void setCDKSwindowLRChar (
CDKSWINDOW ***swindow*,**
chtype *character***);**

void setCDKSwindowPostProcess (
CDKSWINDOW ***swindow*,**
PROCESSFN *callback***,**
void * *data***);**

void setCDKSwindowPreProcess (
CDKSWINDOW ***swindow*,**
PROCESSFN *callback***,**
void * *data***);**

void setCDKSwindowULChar (
CDKSWINDOW ****swindow*,**
chtype *character***);**

void setCDKSwindowURChar (
CDKSWINDOW ****swindow*,**
chtype *character***);**

void setCDKSwindowVerticalChar (
CDKSWINDOW ****swindow*,**
chtype *character***);**

void trimCDKSwindow (
CDKSWINDOW ****swindow*,**
int *start***,**
int *finish***);**

DESCRIPTION

The Cdk scrolling window (swindow) widget can be used to display messages. The following functions create or manipulate the Cdk swindow box widget.

AVAILABLE FUNCTIONS

activateCDKSwindow

function activates the swindow widget and lets the user interact with the widget. The parameter **swindow** is a pointer to a non-NULL swindow 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. 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*.

addCDKSwindow

adds information to a scrolling window. The information is supplied by the **info** parameter. The information is immediately added to the scrolling window. The position of the new information is governed by the value of the parameter **insertPosition**. This parameter may be two values: *TOP* or *BOTTOM*.

cleanCDKSwindow

clears the information from the window.

destroyCDKSwindow

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

drawCDKSwindow

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

dumpCDKSwindow

saves the contents of the scrolling window into the file specified by the **filename** parameter. It returns -1 on failure, and the number of lines saved if the dump was successful.

eraseCDKSwindow

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

execCDKSwindow

allows the user to execute a shell command and have the output of the shell command direct itself to the scrolling window. The **command** parameter is the command to execute. The **insertPosition** parameter tells where the output will be inserted within the scrolling window.

getCDKSwindowBox

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

getCDKSwindowContents

returns the contents of the scrolling window. The parameter **lines** will be set to the number of lines returned.

injectCDKSwindow
injects a single character into the widget. The parameter **swindow** is a pointer to a non-NULL swindow 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. 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*.

jumpToLineCDKSwindow
moves the scrolling window to the given line. The parameter **line** may be an integer or one of the two predefined values *TOP* and *BOTTOM*.

loadCDKSwindowInformation
allows the user to load the contents of a file into the scrolling window. This function is interactive, and will ask for a filename.

moveCDKSwindow
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.

newCDKSwindow
creates a swindow 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*.

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 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.

saveLines
is the number of lines to save before throwing information away.

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

The **shadow**
turns the shadow on or off around this widget.

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

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

saveCDKSwindowInformation
allows the user to save the contents of the scrolling window into a file. This function is interactive, and will ask for a filename.

setCDKSwindow
lets the programmer modify certain elements of an existing swindow widget. The parameter **info** is a *char* ** of the information to set in the scrolling window; **lines** is the number of lines being added. The other parameter names correspond to the same parameter names listed in the **newCDKSwindow** function.

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

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

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

setCDKSwindowBoxAttribute
sets the attribute of the box.

setCDKSwindowContents
lets the programmer modify certain elements of an existing swindow widget. The parameter **info** is a *char* ** of the information to set in the scrolling window; **lines** is the number of lines being added.

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

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

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

setCDKSwindowPostProcess
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)*.

setCDKSwindowPreProcess
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)*.

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

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

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

trimCDKSwindow
removes information from a scrolling window. The parameters **start** and **end** state where to start cutting from and where to stop. The first element in the scrolling window starts at index 0.

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 Right Arrow Up Arrow Down Arrow	Scrolls the window left one column. Scrolls the window right one column. Scrolls the window up one row. Scrolls the window down one row.
Prev Page Ctrl-B b B	Scroll the window backward one page.
Next Page Ctrl-F Space f F	Scroll the window forward one page.
Home |	Scroll the list to the left margin.
End $	Scroll the list to the right margin.
1 < g	Move to the top of the scrolling window.
> G	Move to the bottom of the scrolling window.
l L	Load a file into the scrolling window.
s S	Save the contents of the scrolling window into a file.
Return	Set the widget's *exitType* to *vNORMAL*, exit the widget and return 1.
Tab	Set the widget's *exitType* to *vNORMAL*, exit the widget and return 1.
Escape	Set the widget's *exitType* to *vESCAPE_HIT*, exit the widget and return -1.
Ctrl-L	Refreshes the screen.

SEE ALSO

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