NAME

cdk_binding - **Curses Development Kit** Character Binding Capabilities.

SYNOPSIS

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

#include <cdk.h>

void bindCDKObject (
EObjectType *cdkType***,**
void ***_object_,**
chtype *key***,**
BINDFN *function***,**
void ***_data_);**

int checkCDKObjectBind (
EObjectType *cdkType***,**
void ***_object_,**
chtype *key***);**

void cleanCDKObjectBindings (
EObjectType *cdkType***,**
void ***_object_);**

bool isCDKObjectBind (
EObjectType *cdkType***,**
void ***_object_,**
chtype *key***);**

void unbindCDKObject (
EObjectType *cdkType***,**
void ***_object_,**
chtype *key***);**

int getcCDKObject (
CDKOBJS ***_object_);**

int getchCDKObject (
CDKOBJS ***_object_,**
boolean * *functionKey***);**

DESCRIPTION

Cdk has the ability to create user definable key bindings. This ability makes Cdk more dynamic and usable for a wide variety of tasks. The following section outlines the binding functions, their use, and their purpose.

bindCDKObject

creates a key binding between a specific Cdk widget (**object**) given key (**key**). The parameter **cdkType** is of type *EObjectType* which is one of the following values.

EObjectType_Value	Corresponding_Widget	Widget_Manual_Page
vALPHALIST	Alphalist Widget	cdk_alphalist (3)
vBUTTON	Button Widget	cdk_button (3)
vBUTTONBOX	Buttonbox Widget	cdk_buttonbox (3)
vCALENDAR	Calendar Widget	cdk_calendar (3)
vDIALOG	Dialog Widget	cdk_dialog (3)
vDSCALE	DoubleFloat Widget	cdk_dscale (3)
vENTRY	Entry Widget	cdk_entry (3)
vFSCALE	Floating Scale Widget	cdk_fscale (3)
vFSELECT	File Selector Widget	cdk_fselect (3)
vFSLIDER	Floating Slider Widget	cdk_fslider (3)
vGRAPH	Graph Widget	cdk_graph (3)
vHISTOGRAM	Histogram Widget	cdk_histogram (3)
vITEMLIST	Item List Widget	cdk_itemlist (3)
vLABEL	Label Widget	cdk_label (3)

vMARQUEE	Marquee Widget	cdk_marquee (3)
vMATRIX	Matrix Widget	cdk_matrix (3)
vMENTRY	Multiple Line Entry Widget	cdk_mentry (3)
vMENU	Menu Widget	cdk_menu (3)
vRADIO	Radio List Widget	cdk_radio (3)
vSCALE	Integer Scale Widget	cdk_scale (3)
vSCROLL	Scrolling List Widget	cdk_scroll (3)
vSELECTION	Selection List Widget	cdk_selection (3)
vSLIDER	Slider Widget	cdk_slider (3)
vSWINDOW	Scrolling Window Widget	cdk_swindow (3)
vTEMPLATE	Template Entry Widget	cdk_template (3)
vUSCALE	Unsigned Scale Widget	cdk_uscale (3)
vUSLIDER	Unsigned Slider Widget	cdk_uslider (3)
vVIEWER	Viewer Widget	cdk_viewer (3)

The parameter **function** is the callback function. The parameter **data** points to data passed to the callback function. The parameter **key** is the key hit which triggered this callback.

checkCDKObjectBind
check to see if a binding for the given **key** exists. If it does, Cdk runs the associated command and returns its value, normally TRUE. If no binding exists, return FALSE.

The widgets which accept input, e.g., via "inject" methods, use this to check if the injected character is bound to a function. If that returns TRUE, the widget may update its *exitType* value: if *earlyExit* value is set (not equal to vNEVER_ACTIVATED), the widget sets *exitType* to that value.

cleanCDKObjectBindings
removes all user defined key bindings from the given widget.

isCDKObjectBind
check to see if a binding for the given **key** exists. If it does return TRUE. If no binding exists, return FALSE.

unbindCDKObject
removes a specific binding to an object. The parameter are the same as for **bindCDKObject**.

getcCDKObject
reads a keycode from the given widget. This is depcrecated: use **getchCDKObject**.

getchCDKObject
reads a keycode from the given widget. It sets a flag to indicate if the result is a function key. If the keycode has been bound to the special function **getcCDKBind**, then it will be translated to the value which was given for the binding data. Otherwise, a few special cases are performed:

Key	Result
CTRL-A	KEY_HOME
CTRL-B	KEY_LEFT
CTRL-E	KEY_END
CTRL-F	KEY_RIGHT
CTRL-N	tab
CTRL-P	KEY_BTAB
DEL	KEY_DC
backspace	KEY_BACKSPACE
carriage return	KEY_ENTER
newline	KEY_ENTER

All of the widgets use **getchCDKObject** internally for consistency.

EXAMPLE

To help demonstrate how to use the key bindings I will demonstrate a simple dialog box widget with help for each button. The following code segment creates a dialog box and a callback function named *dialogHelpCB*.

```
#include <cdk.h>

#ifdef HAVE_XCURSES
char *XCursesProgramName="bind_ex";
#endif

static int dialogHelpCB (EObjectType cdktype GCC_UNUSED, void *object, void *clientData GCC_UNUSED, chtype k
{
  CDKDIALOG *dialog = (CDKDIALOG *)object;
  char *mesg[5];

  /* Check which button we are on. */
  if (dialog->currentButton == 0)
  {
    mesg[0] = "<C></U>Help for </U>Who<!U>.";
    mesg[1] = "<C>When this button is picked the name of the current";
    mesg[2] = "<C>user is displayed on the screen in a popup window.";
    popupLabel (ScreenOf(dialog), mesg, 3);
  }
  else if (dialog->currentButton == 1)
  {
    mesg[0] = "<C></U>Help for </U>Time<!U>.";
    mesg[1] = "<C>When this button is picked the current time is";
    mesg[2] = "<C>displayed on the screen in a popup window.";
    popupLabel (ScreenOf(dialog), mesg, 3);
  }
  else if (dialog->currentButton == 2)
  {
    mesg[0] = "<C></U>Help for </U>Date<!U>.";
    mesg[1] = "<C>When this button is picked the current date is";
    mesg[2] = "<C>displayed on the screen in a popup window.";
    popupLabel (ScreenOf(dialog), mesg, 3);
  }
  else if (dialog->currentButton == 3)
  {
    mesg[0] = "<C></U>Help for </U>Quit<!U>.";
    mesg[1] = "<C>When this button is picked the dialog box is exited.";
    popupLabel (ScreenOf(dialog), mesg, 2);
  }
  return (FALSE);
}

int main (void)
{
  /* Declare variables. */
  CDKSCREEN       *cdkscreen;
  CDKDIALOG       *question;
  char                            *buttons[40];
  char                            *message[40], *info[5], *loginName;
  char                            temp[256];
  int                             selection;
  time_t          clck;
  struct tm       *currentTime;

  cdkscreen = initCDKScreen (NULL);

  /* Start color. */
  initCDKColor();
```

```
/* Set up the dialog box. */
message[0] = "<C></U>Simple Command Interface";
message[1] = "Pick the command you wish to run.";
message[2] = "<C>Press </R>?<!R> for help.";
buttons[0] = "Who";
buttons[1] = "Time";
buttons[2] = "Date";
buttons[3] = "Quit";

/* Create the dialog box. */
question                = newCDKDialog (cdkscreen, CENTER, CENTER,
                                                                        message, 3, buttons, 4, A_REVERSE,
                                                                        TRUE, TRUE, FALSE);

/* Check if we got a null value back. */
if (question == (CDKDIALOG *)0)
{
  destroyCDKScreen (cdkscreen);

  /* End curses... */
  endCDK();

  /* Spit out a message. */
  printf ("Oops. Can't seem to create the dialog box. Is the window too small?\n");
  exit (1);
}

/* Create the key binding. */
bindCDKObject (vDIALOG, question, '?', dialogHelpCB, 0);

/* Activate the dialog box. */
selection = 0;
while (selection != 3)
{
  /* Get the users button selection. */
  selection = activateCDKDialog (question, (chtype *)0);

  /* Check the results. */
  if (selection == 0)
  {
                        /* Get the users login name. */
                        info[0] = "<C>     </U>Login Name<!U>     ";
                        loginName = getlogin();
                        if (loginName == (char *)0)
                        {
                          strcpy (temp, "<C></R>Unknown");
                        }
                        else
                        {
                           sprintf (temp, "<C><%s>", loginName);
                        }
                        info[1] = copyChar (temp);
                        popupLabel (ScreenOf(question), info, 2);
                        freeChar (info[1]);
  }
  else if (selection == 1)
  {
                        /* Print out the time. */
                        time(&clck);
```

```
                        currentTime = localtime(&clck);
                        sprintf (temp, "<C>%d:%d:%d", currentTime->tm_hour,
                                                                currentTime->tm_m
                                                                currentTime->tm_s

                        info[0] = "<C>   </U>Current Time<!U>   ";
                        info[1] = copyChar (temp);
                        popupLabel (ScreenOf(question), info, 2);
                        freeChar (info[1]);
    }
    else if (selection == 2)
    {
                        /* Print out the date. */
                        time(&clck);
                        currentTime = localtime(&clck);
                        sprintf (temp, "<C>%d/%d/%02d", currentTime->tm_mday,
                                                                currentTime->tm_m
                                                                currentTime->tm_y

                        info[0] = "<C>   </U>Current Date<!U>   ";
                        info[1] = copyChar (temp);
                        popupLabel (ScreenOf(question), info, 2);
                        freeChar (info[1]);
    }
  }

  /* Clean up. */
  destroyCDKDialog (question);
  destroyCDKScreen (cdkscreen);
  endCDK();
  exit (0);
}
```

SEE ALSO

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