NAME

checkbutton − Create and manipulate 'checkbutton' boolean selection widgets

SYNOPSIS

checkbutton *pathName* ?*options*?

STANDARD OPTIONS

−activebackground	**−disabledforeground**	**−padx**
−activeforeground	**−font**	**−pady**
−anchor	**−foreground**	**−relief**
−background	**−highlightbackground**	**−takefocus**
−bitmap	**−highlightcolor**	**−text**
−borderwidth	**−highlightthickness**	**−textvariable**
−compound	**−image**	**−underline**
−cursor	**−justify**	**−wraplength**

See the **options** manual entry for details on the standard options.

WIDGET-SPECIFIC OPTIONS

Command-Line Name: **−command**
Database Name: **command**
Database Class: **Command**

Specifies a Tcl command to associate with the button. This command is typically invoked when mouse button 1 is released over the button window. The button's global variable (**−variable** option) will be updated before the command is invoked.

Command-Line Name: **−height**
Database Name: **height**
Database Class: **Height**

Specifies a desired height for the button. If an image or bitmap is being displayed in the button then the value is in screen units (i.e. any of the forms acceptable to **Tk_GetPixels**); for text it is in lines of text. If this option is not specified, the button's desired height is computed from the size of the image or bitmap or text being displayed in it.

Command-Line Name: **−indicatoron**
Database Name: **indicatorOn**
Database Class: **IndicatorOn**

Specifies whether or not the indicator should be drawn. Must be a proper boolean value. If false, the **−relief** option is ignored and the widget's relief is always sunken if the widget is selected and raised otherwise.

Command-Line Name: **−offrelief**
Database Name: **offRelief**
Database Class: **OffRelief**

Specifies the relief for the checkbutton when the indicator is not drawn and the checkbutton is off. The default value is "raised". By setting this option to "flat" and setting **−indicatoron** to false and **−overrelief** to "raised", the effect is achieved of having a flat button that raises on mouse-over and which is depressed when activated. This is the behavior typically exhibited by the Bold, Italic, and Underline checkbuttons on the toolbar of a word-processor, for example.

Command-Line Name: **−offvalue**
Database Name: **offValue**
Database Class: **Value**

Specifies value to store in the button's associated variable whenever this button is deselected. Defaults to "0".

Command-Line Name: **−onvalue**
Database Name: **onValue**
Database Class: **V alue**

Specifies value to store in the button's associated variable whenever this button is selected. Defaults to "1".

Command-Line Name: **−o verrelief**
Database Name: **overRelief**
Database Class: **Ov erRelief**

Specifies an alternative relief for the checkbutton, to be used when the mouse cursor is over the widget. This option can be used to make toolbar buttons, by configuring **−relief flat −overrelief raised**. If the value of this option is the empty string, then no alternative relief is used when the mouse cursor is over the checkbutton. The empty string is the default value.

Command-Line Name: **−selectcolor**
Database Name: **selectColor**
Database Class: **Backgr ound**

Specifies a background color to use when the button is selected. If **indicatorOn** is true then the color is used as the background for the indicator regardless of the select state. If **indicatorOn** is false, this color is used as the background for the entire widget, in place of **background** or **activeBackground**, whenever the widget is selected. If specified as an empty string then no special color is used for displaying when the widget is selected.

Command-Line Name: **−selectimage**
Database Name: **selectImage**
Database Class: **SelectImage**

Specifies an image to display (in place of the **−image** option) when the checkbutton is selected. This option is ignored unless the **−image** option has been specified.

Command-Line Name: **−state**
Database Name: **state**
Database Class: **State**

Specifies one of three states for the checkbutton: **normal**, **active**, or **disabled**. In normal state the checkbutton is displayed using the **−foreground** and **−background** options. The active state is typically used when the pointer is over the checkbutton. In active state the checkbutton is displayed using the **−activeforeground** and **−activebackground** options. Disabled state means that the checkbutton should be insensitive: the default bindings will refuse to activate the widget and will ignore mouse button presses. In this state the **−disabledforeground** and **−background** options determine how the checkbutton is displayed.

Command-Line Name: **−tristateimage**
Database Name: **tristateImage**
Database Class: **T ristateImage**

Specifies an image to display (in place of the **−image** option) when the checkbutton is in tri-state mode. This option is ignored unless the**−image** option has been specified.

Command-Line Name: **−tristatevalue**
Database Name: **tristateValue**
Database Class: **V alue**

Specifies the value that causes the checkbutton to display the multi-value selection, also known as the tri-state mode. Defaults to "".

Command-Line Name: **−variable**
Database Name: **variable**

Database Class: **V ariable**

Specifies the name of a global variable to set to indicate whether or not this button is selected. Defaults to the name of the button within its parent (i.e. the last element of the button window's path name).

Command-Line Name: **−width**
Database Name: **width**
Database Class: **W idth**

Specifies a desired width for the button. If an image or bitmap is being displayed in the button then the value is in screen units (i.e. any of the forms acceptable to **Tk_GetPixels**); for text it is in characters. If this option is not specified, the button's desired width is computed from the size of the image or bitmap or text being displayed in it.

DESCRIPTION

The **checkbutton** command creates a new window (given by the *pathName* argument) and makes it into a checkbutton widget. Additional options, described above, may be specified on the command line or in the option database to configure aspects of the checkbutton such as its colors, font, text, and initial relief. The **checkbutton** command returns its *pathName* argument. At the time this command is invoked, there must not exist a window named *pathName*, but *pathName*'s parent must exist.

A checkbutton is a widget that displays a textual string, bitmap or image and a square called an *indicator*. If text is displayed, it must all be in a single font, but it can occupy multiple lines on the screen (if it contains newlines or if wrapping occurs because of the **−wraplength** option) and one of the characters may optionally be underlined using the **−underline** option. A checkbutton has all of the behavior of a simple button, including the following: it can display itself in either of three different ways, according to the **−state** option; it can be made to appear raised, sunken, or flat; it can be made to flash; and it invokes a Tcl command whenever mouse button 1 is clicked over the checkbutton.

In addition, checkbuttons can be *selected*. If a checkbutton is selected then the indicator is normally drawn with a selected appearance, and a Tcl variable associated with the checkbutton is set to a particular value (normally 1). The indicator is drawn with a check mark inside. If the checkbutton is not selected, then the indicator is drawn with a deselected appearance, and the associated variable is set to a different value (typically 0). The indicator is drawn without a check mark inside. In the special case where the variable (if specified) has a value that matches the tristatevalue, the indicator is drawn with a tri-state appearance and is in the tri-state mode indicating mixed or multiple values. (This is used when the check box represents the state of multiple items.) The indicator is drawn in a platform dependent manner. Under Unix and Windows, the background interior of the box is "grayed". Under Mac, the indicator is drawn with a dash mark inside. By default, the name of the variable associated with a checkbutton is the same as the *name* used to create the checkbutton. The variable name, and the "on", "off" and "tristate" values stored in it, may be modified with options on the command line or in the option database. Configuration options may also be used to modify the way the indicator is displayed (or whether it is displayed at all). By default a checkbutton is configured to select and deselect itself on alternate button clicks. In addition, each checkbutton monitors its associated variable and automatically selects and deselects itself when the variables value changes to and from the button's "on", "off" and "tristate" values.

WIDGET COMMAND

The **checkbutton** command creates a new Tcl command whose name is *pathName*. This command may be used to invoke various operations on the widget. It has the following general form:

pathName option ?*arg arg ...*?

Option and the *arg*s determine the exact behavior of the command. The following commands are possible for checkbutton widgets:

pathName **cget** *option*
Returns the current value of the configuration option given by *option*. *Option* may have any of the values accepted by the **checkbutton** command.

pathName **configure** ?*option*? ?*value option value ...*?
Query or modify the configuration options of the widget. If no *option* is specified, returns a list describing all of the available options for *pathName* (see **Tk_ConfigureInfo** for information on the format of this list). If *option* is specified with no *value*, then the command returns a list describing the one named option (this list will be identical to the corresponding sublist of the value returned if no *option* is specified). If one or more *option−value* pairs are specified, then the command modifies the given widget option(s) to have the given value(s); in this case the command returns an empty string. *Option* may have any of the values accepted by the **checkbutton** command.

pathName **deselect**
Deselects the checkbutton and sets the associated variable to its "off" value.

pathName **flash**
Flashes the checkbutton. This is accomplished by redisplaying the checkbutton several times, alternating between active and normal colors. At the end of the flash the checkbutton is left in the same normal/active state as when the command was invoked. This command is ignored if the checkbutton's state is **disabled**.

pathName **invoke**
Does just what would have happened if the user invoked the checkbutton with the mouse: toggle the selection state of the button and invoke the Tcl command associated with the checkbutton, if there is one. The return value is the return value from the Tcl command, or an empty string if there is no command associated with the checkbutton. This command is ignored if the checkbutton's state is **disabled**.

pathName **select**
Selects the checkbutton and sets the associated variable to its "on" value.

pathName **toggle**
Toggles the selection state of the button, redisplaying it and modifying its associated variable to reflect the new state.

BINDINGS

Tk automatically creates class bindings for checkbuttons that give them the following default behavior:

[1] On Unix systems, a checkbutton activates whenever the mouse passes over it and deactivates whenever the mouse leaves the checkbutton. On Mac and Windows systems, when mouse button 1 is pressed over a checkbutton, the button activates whenever the mouse pointer is inside the button, and deactivates whenever the mouse pointer leaves the button.

[2] When mouse button 1 is pressed over a checkbutton, it is invoked (its selection state toggles and the command associated with the button is invoked, if there is one).

[3] When a checkbutton has the input focus, the space key causes the checkbutton to be invoked. Under Windows, there are additional key bindings; plus (+) and equal (=) select the button, and minus (−) deselects the button.

If the checkbutton's state is **disabled** then none of the above actions occur: the checkbutton is completely non-responsive.

The behavior of checkbuttons can be changed by defining new bindings for individual widgets or by redefining the class bindings.

EXAMPLE

This example shows a group of uncoupled checkbuttons.

```
labelframe .lbl −text "Steps:"
```

```
checkbutton .c1 −text Lights  −variable lights
checkbutton .c2 −text Cameras −variable cameras
checkbutton .c3 −text Action! −variable action
pack .c1 .c2 .c3 −in .lbl
pack .lbl
```

SEE ALSO

button(n), options(n), radiobutton(n), ttk::checkbutton(n)

KEYWORDS

checkbutton, widget