**Example**

```
bl := blinklabel(spareframe, 'flashdance');
bl.blink(T, 0.2); # blink every 0.2 seconds
# ... some time passes ...
bl.blink(F);      # stop blinking
```

---

**blinklabel.done - Function**

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **blinklabel** |

*Destroy ourselves*

**Synopsis**
*done()*

**Returns**
*Bool*

**Description**

*This tool function destroys the widget.*

**Example**

```
- bl := blinklabel(spareframe, 'flashdance');
- bl.done();
T
- bl
F
-
```

---

### 1.4.6 checkmenu - Tool

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |

*Tk widget to provide a flat check button menu*
*include "checkmenu.g"*

### Constructors

| | |
|---|---|
| *checkmenu* | *Construct a checkmenu* |

### Functions

| | |
|---|---|
| *disabled* | *Disable or enable the checkmenu widget* |
| *extend* | *Extend the menu with more items* |
| *findname* | *Find the index of the item with the given name* |
| *findvalue* | *Find the index of the item with the given value* |
| *getlabel* | *Get the label of the menu button* |
| *getnames* | *Gets all of the names associated with the menu* |
| *getoffstates* | *Get the indices of the items which are not checked* |
| *getonstates* | *Get the indices of the items which are checked* |
| *getstate* | *Get the state of the specified check button* |
| *getstates* | *Get the states of all of the check buttons* |
| *getvalues* | *Gets all of the values associated with the menu* |
| *isenabled* | *Is the checkmenu widget enabled ?* |
| *replace* | *Replace the menu with the provided items* |
| *reset* | *Resets the state of all check buttons to F* |
| *selectindex* | *Select an item in the menu by index* |
| *selectname* | *Select an item in the menu by name* |
| *selectvalue* | *Select an item in the menu by value* |
| *setbackground* | *Set the color of the background of the checkmenu button* |
| *setforeground* | *Set the color of the foreground of the checkmenu button* |
| *setlabel* | *Set the label of the checkmenu* |

### Description

*This Tk widget enables you to create a flat (i.e. no submenus) menu in a specified frame where all the menu items are check buttons. You can associate a name (that which appears in the menu list) and a value (whatever you like, it may be different from the name) for each menu item.*
*It can be treated like any of the built in Tk widgets, and has almost all of the same construction arguments as the button Tk widget.*
*Whenever a selection is made, this widget emits an event called* `select`. *In the $value of this event, you can find a record with fields* `name`, `value`, `state` *(T or F) and* `index` *(index of item in menu) containing that information for the selected item.*
*Whenever the menu is replaced with the replace tool function, this widget emits an event called* `replaced` *(it has no value). Note that the tool function extend invokes the* `replace` *function, so a* `replaced` *event is generated then as well.*

## checkmenu - Constructor

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Construct a checkmenu*

**Synopsis**
*checkmenu(parent, label, names, values, hlp, hlp2, padx, pady, width, height, justify, font, relief, borderwidth, foreground, background, disabled, anchor, fill, widgetset)*

**Arguments**

| | |
|---|---|
| *parent* | *Parent of the menu*<br>*Allowed: Widget* |
| *label* | *The label of the menu*<br>*Allowed: String*<br>*Default: "checkmenu"* |
| *names* | *The menu item names*<br>*Allowed: Vector of strings*<br>*Default: Empty menu* |
| *values* | *The menu item values*<br>*Allowed: Vector of anything*<br>*Default: Empty menu* |
| *hlp* | *Short popuphelp text*<br>*Allowed: String*<br>*Default: No help* |
| *hlp2* | *Long popuphelp text*<br>*Allowed: String*<br>*Default: No help* |
| *padx* | *Horizontal padding around text*<br>*Allowed: Integer*<br>*Default: 7* |
| *pady* | *Vertical padding around text*<br>*Allowed: Integer*<br>*Default: 3* |
| *width* | *Width in character units*<br>*Allowed: Integer*<br>*Default: 0 [Auto-scale]* |
| *height* | *Height in character units*<br>*Allowed: Integer*<br>*Default: 1* |
| *justify* | *Justification of text*<br>*Allowed: String from 'center', 'left', 'right'*<br>*Default: 'center'* |
| *font* | *Font of text*<br>*Allowed: X font* |
| *relief* | *Edge relief*<br>*Allowed: String from 'flat', 'ridge', 'raised', 'sunken', 'groove'*<br>*Default: 'groove'* |
| *borderwidth* | *Border width*<br>*Allowed: Integer*<br>*Default: 2* |
| *foreground* | *Color of text*<br>*Allowed: X color*<br>*Default: 'black'* |
| *background* | *Background color*<br>*Allowed: X color*<br>*Default: 'lightgrey'* |
| *disabled* | *Is inactivated ?*<br>*Allowed: Boolean*<br>*Default: F* |
| *anchor* | *Location of text*<br>*Allowed: String from 'c', 'n', 's', 'e', 'w', 'ne', 'nw', 'se', 'sw'*<br>*Default: 'c'* |
| *fill* | *How to expand when resized*<br>*Allowed: String from 'x', 'y', 'both', 'none'*<br>*Default: 'none'* |
| *widgetset* | *The widget server to use*<br>*Allowed: widgetserver tool*<br>*Default: dws* |

**Returns**
*Agent*

**Description**

**Example**

```
- include 'checkmenu.g'
- f := frame();
- hlp := 'Select type of model to fit';
- m := checkmenu(f, 'Models', "gaussian disk point", [1,2,3], hlp);
```

*The values vector, if not given, defaults to the names vector.*

---

### checkmenu.extend - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Extend the menu with more items*

**Synopsis**
*extend(names, values)*

**Arguments**

| | |
|---|---|
| *names* | *The menu item names*<br>*Allowed: Vector of strings*<br>*Default: Empty menu* |
| *values* | *The menu item values*<br>*Allowed: Vector of anything*<br>*Default: Empty menu* |

**Returns**
*Bool*

**Description**

*This tool function extends the menu with the supplied items.*

### Example

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian point");
- m.getnames();
gaussian point
- m.extend(names="disk")
- m.getnames();
gaussian point disk
-
```

---

## checkmenu.replace - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Replace the menu with the provided items*

### Synopsis

*replace(label, names, values, width)*

### Arguments

| | |
|---|---|
| *label* | *The label of the menu*<br>*Allowed: String*<br>*Default: "checkmenu"* |
| *names* | *The menu item names*<br>*Allowed: Vector of strings*<br>*Default: Empty menu* |
| *values* | *The menu item values*<br>*Allowed: Vector of anything*<br>*Default: Empty menu* |
| *width* | *Width in character units*<br>*Allowed: Integer*<br>*Default: 0 [Auto-scale]* |

### Returns

*Bool*

### Description

*This tool function replaces the menu with the supplied items. If you set* `width=-1` *then the menu button width will be left as it is. Thus if you constructed with* `width=0`*, the button will resize itself to accomodate the new text. If you used a specific non-zero width at construction time, setting* `width=-1` *now will leave the button at that width. Setting* `width=0` *will now force a resize and setting* `width` *to some positive integer will resize to that width.*

### Example

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian point", [1,2])
- m.getnames();
gaussian point
- m.getvalues();
[1 2]
- m.replace('Animals', "cat dog fish");
- m.getnames();
cat dog fish
- m.getvalues();
cat dog fish
-
```

---

## checkmenu.selectindex - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Select an item in the menu by index*

### Synopsis
*selectindex(idx, state)*

### Arguments

| | | |
|---|---|---|
| *idx* | *The menu item index*<br>*Allowed: Integer* |
| *state* | *The state for the check box*<br>*Allowed: T of F*<br>*Default: Opposite state* |

**Returns**
*Bool*

**Description**

*This tool function selects a menu item by its index. It is equivalent to selecting the item with the GUI interface. If the index is invalid, it returns* `F`*. If you don't specify the* `state` *variable, the check button is given the opposite state to its current state.*

**Example**

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian point");
- m.getstates()
[F F]
- m.selectindex(2);
- m.getstates()
[F T]
- m.selectindex(2, T);
[F T]
-
```

---

**checkmenu.selectname - Function**

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Select an item in the menu by name*

**Synopsis**
*selectname(name, state)*

### Arguments

| | | |
|---|---|---|
| *name* | *The menu item name*<br>*Allowed: String* |
| *state* | *The state for the check box*<br>*Allowed: T of F*<br>*Default: Opposite state* |

### Returns
*Bool*

### Description

*This tool function selects a menu item by its name. It is equivalent to selecting the item with the GUI interface. If the name is invalid, it returns* `F`*. If you don't specify the* `state` *variable, the check button is given the opposite state to its current state.*

### Example

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian point");
- m.getstates()
[F F]
- m.selectname('gaussian');
- m.getstates()
[T F]
- m.selectname('gaussian', T);
[T F]
-
```

---

## checkmenu.selectvalue - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Select an item in the menu by value*

### Synopsis

*selectvalue(value, state)*

### Arguments

| | |
|---|---|
| *value* | *The menu item value*<br>*Allowed: Same type as at construction* |
| *state* | *The state for the check box*<br>*Allowed: T of F*<br>*Default: Opposite state* |

### Returns

*Bool*

### Description

*This tool function selects a menu item by its value. It is equivalent to selecting the item with the GUI interface. If the value is invalid, it returns* `F`*. If you don't specify the* `state` *variable, the check button is given the opposite state to its current state.*

### Example

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian point", [10,20])
- m.getstates()
[F F]
- m.selectvalue(10)
- m.getstates()
[T F]
- m.selectvalue(10, T);
[T F]
-
```

---

## checkmenu.getlabel - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Get the label of the menu button*

**Synopsis**
*getlabel()*

**Returns**
*String or F*

**Description**

*This tool function recovers the label of the menu button.*

**Example**

```
- f := frame();
- m := checkmenu(f);
- m.getlabel();
checkmenu
```

---

**checkmenu.getnames - Function**

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Gets all of the names associated with the menu*

**Synopsis**
*getnames()*

**Returns**
*Vector of Strings*

**Description**

*This tool function recovers the names associated with each item in the menu. If the menu is empty, it returns an empty array.*

### Example

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian point", [1,2])
- m.getnames();
gaussian point
```

---

## checkmenu.getvalues - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Gets all of the values associated with the menu*

### Synopsis
*getvalues()*

### Returns
*The "values" array given (explicitly or implicitly) in the constructor*

### Description

*This tool function recovers the values associated with each item in the menu. If the menu is empty, it returns an empty array.*

### Example

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian point", [1,2])
- m.getvalues()
[1 2]
-
```

---

## checkmenu.getstate - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Get the state of the specified check button*

**Synopsis**
*getstate(idx)*

**Arguments**

| | |
|---|---|
| *idx* | *The index of the menu item*<br>*Allowed: Integer* |

**Returns**
*T or F or fail*

**Description**

*This tool function recovers the state of the specified check button. A fail is generated for an empty checkmenu or invalid index.*

**Example**

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian disk");
- m.getstate(1);
F
```

---

### checkmenu.getstates - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Get the states of all of the check buttons*

**Synopsis**
*getstates()*

**Returns**

*T or F*

### Description

*This tool function recovers the states of all of the check buttons.*

### Example

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian disk");
- m.getstates();
[F F]
```

---

## checkmenu.getonstates - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Get the indices of the items which are checked*

### Synopsis
*getonstates()*

### Returns
*Vector of integers*

### Description

*This tool function recovers the indices of the check menu items which are checked.*

### Example

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian disk point");
- m.selectindex(1,T);
```

```
- m.selectindex(3,T);
- m.getstates();
[T F T]
- m.getonstates();
[1 3]
```

---

### checkmenu.getoffstates - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Get the indices of the items which are not checked*

**Synopsis**
*getoffstates()*

**Returns**
*Vector of integers*

**Description**

*This tool function recovers the indices of the check menu items which are not checked.*

**Example**

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian disk point");
- m.selectindex(1,T);
- m.selectindex(3,T);
- m.getstates();
[T F T]
- m.getoffstates();
[2]
```

---

### checkmenu.findname - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Find the index of the item with the given name*

**Synopsis**
*findname(name)*

**Arguments**

| | | |
|---|---|---|
| *name* | *The menu item name* <br> *Allowed: String* | |

**Returns**
*Integer or F*

**Description**

*This tool function find the index in the menu of the item specified by its name. If you give an invalid name,* `F` *is returned.*

**Example**

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian point");
- m.findname('point');
2
```

---

**checkmenu.findvalue - Function**

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Find the index of the item with the given value*

**Synopsis**
*findvalue(value)*

**Arguments**

| | |
|---|---|
| *value* | *The menu item value*<br>*Allowed: Same type as the array "value" used in construction* |

**Returns**
*Integer or F*

**Description**

*This tool function find the index in the menu of the item specified by its value. If you give an invalid value,* F *is returned.*

**Example**

```
- f := frame();
- m := checkmenu(f, 'Models', "gaussian point", [10,20]);
- m.findvalue(20);
2
-
```

---

### checkmenu.setlabel - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Set the label of the checkmenu*

**Synopsis**
*setlabel(label, width)*

**Arguments**

| | |
|---|---|
| *label* | *The label*<br>*Allowed: String* |
| *width* | *Width of button*<br>*Allowed: Integer*<br>*Default: -1* |

**Returns**
*Bool*

**Description**

*This tool function sets the label of the checkmenu. If you set* `width=-1` *then the menu button width will be left as it is. Thus if you constructed with* `width=0`*, the button will resize itself to accomodate the new text. If you used a specific non-zero width at construction time, setting* `width=-1` *now will leave the button at that width. Setting* `width=0` *will now force a resize and setting* `width` *to some positive integer will resize to that width.*

**Example**

```
- f := frame();
- m := checkmenu(f, 'Animals', "dog cat");
- m.getlabel()
Animals
- m.setlabel('Small animals');
- m.getlabel()
Small animals
-
```

---

## checkmenu.reset - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Resets the state of all check buttons to F*

**Synopsis**
*reset()*

**Returns**
*Bool*

**Description**

*This tool function resets the state of all check buttons to F.*

---

### checkmenu.setforeground - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Set the color of the foreground of the checkmenu button*

**Synopsis**
*setforeground(color)*

**Arguments**

| | |
|---|---|
| *color* | *The color*<br>*Allowed: Any X color encoded as a String* |

**Returns**
*Bool*

**Description**

*This tool function sets the color of the foreground of the checkmenu button.*

---

### checkmenu.setbackground - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Set the color of the background of the checkmenu button*

**Synopsis**
*setbackground(color)*

**Arguments**

| | |
|---|---|
| *color* | *The color*<br>*Allowed: Any X color encoded as a String* |

**Returns**
*Bool*

**Description**

*This tool function sets the color of the background of the checkmenu button.*

---

### checkmenu.disabled - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |

*Disable or enable the checkmenu widget*

**Synopsis**
*disabled(disable)*

**Arguments**

| | |
|---|---|
| *disable* | *Disable the widget*<br>*Allowed: T or F*<br>*Default: T* |

**Returns**
*Bool*

**Description**

*This tool function disables or enables the checkmenu.*

---

### checkmenu.isenabled - Function

| | |
|---|---|
| **Package** | **display** |
| **Module** | **widgets** |
| **Tool** | **checkmenu** |