

---

# Galaxie Curses

*Release 0.3.5*

Jun 05, 2023



---

## Contents

---

<b>1</b>	<b>The Project</b>	<b>3</b>
<b>2</b>	<b>The Mission</b>	<b>5</b>
<b>3</b>	<b>Example</b>	<b>7</b>
<b>4</b>	<b>Features</b>	<b>9</b>
<b>5</b>	<b>Contribute</b>	<b>11</b>
<b>6</b>	<b>Documentations</b>	<b>13</b>
6.1	Installation . . . . .	13
6.1.1	Next Step: . . . . .	14
6.2	GLXCurses . . . . .	14
6.2.1	GLXCurses package . . . . .	14
6.3	Basic Types . . . . .	213
6.4	Constant Value . . . . .	213
6.4.1	enum BaselinePosition . . . . .	213
6.4.2	enum DeleteType . . . . .	214
6.4.3	enum DirectionType . . . . .	214
6.4.4	enum Justification . . . . .	214
6.4.5	enum MovementStep . . . . .	215
6.4.6	enum Orientation . . . . .	215
6.4.7	enum PackType . . . . .	215
6.4.8	enum PositionType . . . . .	215
6.4.9	enum ReliefStyle . . . . .	216
6.4.10	enum ScrollStep . . . . .	216
6.4.11	enum ScrollType . . . . .	216
6.4.12	enum SelectionMode . . . . .	217
6.4.13	enum ShadowType . . . . .	217
6.4.14	enum StateFlags . . . . .	217
6.4.15	enum ToolbarStyle . . . . .	218
6.4.16	enum SortType . . . . .	218
6.5	Contributors . . . . .	218
<b>7</b>	<b>Note for GTK+ Project Developer's</b>	<b>219</b>

<b>8</b>	<b>License</b>	<b>221</b>
<b>9</b>	<b>Indices and tables</b>	<b>223</b>
	<b>Python Module Index</b>	<b>225</b>
	<b>Index</b>	<b>227</b>



Galaxie Curses is being developed on **<https://gitlab.com>** , a development platform that is itself part of the open source community and that can be self-hosted, if the need arises.

Actual Homepage : **<https://gitlab.com/Tuuux/galaxie-curses>**

Previous homepage: “<https://github.com/Tuuux/galaxie-curses>”.



# CHAPTER 1

---

## The Project

---

**Galaxie Curses** is a free software ToolKit for the NCurses API. It can be consider as a text based implementation of the famous GTK+ Library.

Originally the project have start in 2016 by the author Jérôme.O Alias Tuuux.





## CHAPTER 2

---

### The Mission

---

Provide a Text Based ToolKit with powerfull high level Widget (Select Color, Printer Dialog, FileSelector).

During lot of years the main stream was to provide big computer with big GUI Toolkit, unfortunately almost nobody have care about ultra low profile computer and we are now in a situation where no mature ToolKit is ready to use on **pen computer**. Time's change then it's time to change the world ...

The goal of the version 1.0 will be to create a application like [Midnight-Commander](#) with **GLXCurses**.



## CHAPTER 3

---

### Example

---

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import GLXCurses

# Create the main Application
app = GLXCurses.Application()
app.set_name('Galaxie-Curse Container Demo')

# Create a Window
win_main = GLXCurses.Window()

# Create a Frame
frame1 = GLXCurses.Frame()
frame1.set_label('A Container Frame')

# Add the frame to the main window
win_main.add(frame1)

# Add the main window inside the application
app.add_window(win_main)

# The super function call when press keys
def handle_keys(self, event_signal, *event_args):
    if event_args[0] == ord('q'):
        # Key "q" was pressed
        GLXCurses.mainloop.quit()

# Signal
app.connect('CURSES', handle_keys)

# Main loop start
GLXCurses.mainloop.run()
```

More examples can be found here: <https://gitlab.com/Tuuux/galaxie-curses/tree/master/examples>



## CHAPTER 4

---

### Features

---

- MainLoop
- Signal
- Application Class
- Component like Button, Container, ProgressBar
- Have GTK+ design as roadmap
- Auto Resize
- Minimize NCurses crash
- Common thing for a text based graphic interface tool kit :)



## CHAPTER 5

---

### Contribute

---

The GTK+ documentation is our model: <https://developer.gnome.org/gtk3/stable/>

- Issue Tracker: <https://gitlab.com/Tuuux/galaxie-curses/issues>
- Source Code: <https://gitlab.com/Tuuux/galaxie-curses>

Our collaboration model is the Collective Code Construction Contract (C4): <https://rfc.zeromq.org/spec:22/C4/>





### 6.1 Installation

You can found the Galaxie Curses Repository here: <https://gitlab.com/Tuuux/galaxie-curses>

In any case it consist to copy the package GLXCurses inside you developing project directory.

Before you start, make sure that you already have installed **Python**, **pip** and **git**.

- Then clone Galaxie Curses project from Gitlab:

```
git clone https://gitlab.com/Tuuux/galaxie-curses.git
```

It will create a folder name `galaxie-curses` it contain the GLXCurses package:

- Create a directory for you application project:

```
mkdir ./SuperApplication
```

- Enter inside SuperApplication :

```
cd ./SuperApplication
```

- Move the GLXCurses directory inside the SuperApplication folder:

```
mv ../galaxie-curses/GLXCurses ./
```

Now you can import the GLXCuses package

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import GLXCurses
```

### 6.1.1 Next Step:

- Take a look on our example's files
- Enjoy ;-)

## 6.2 GLXCurses

### 6.2.1 GLXCurses package

#### Subpackages

#### GLXCurses.libs package

#### Subpackages

#### GLXCurses.libs.handlers package

#### Submodules

#### GLXCurses.libs.handlers.application module

```
class GLXCurses.libs.handlers.application.HandlersApplication  
    Bases: object
```

#### GLXCurses.libs.handlers.button module

```
class GLXCurses.libs.handlers.button.HandlersButton  
    Bases: object
```

#### GLXCurses.libs.handlers.container module

```
class GLXCurses.libs.handlers.container.HandlersContainer  
    Bases: object
```

#### GLXCurses.libs.handlers.editable module

```
class GLXCurses.libs.handlers.editable.HandlersEditable  
    Bases: object
```

#### GLXCurses.libs.handlers.filechooser module

```
class GLXCurses.libs.handlers.filechooser.HandlersFileChooser  
    Bases: object
```

### GLXCurses.libs.handlers.label module

```
class GLXCurses.libs.handlers.label.HandlersLabel
    Bases: object
```

### GLXCurses.libs.handlers.statusbar module

```
class GLXCurses.libs.handlers.statusbar.HandlersStatusbar
    Bases: object
```

### GLXCurses.libs.handlers.textview module

```
class GLXCurses.libs.handlers.textview.HandlersTextView
    Bases: object
```

### GLXCurses.libs.handlers.widget module

```
class GLXCurses.libs.handlers.widget.HandlersWidget
    Bases: object
```

### GLXCurses.libs.handlers.window module

```
class GLXCurses.libs.handlers.window.HandlersWindow
    Bases: object
```

## Module contents

### Submodules

### GLXCurses.libs.ApplicationHandlers module

```
class GLXCurses.libs.ApplicationHandlers.Handlers
    Bases: GLXCurses.libs.handlers.application.HandlersApplication, GLXCurses.
libs.handlers.button.HandlersButton, GLXCurses.libs.handlers.container.
HandlersContainer, GLXCurses.libs.handlers.editable.HandlersEditable,
GLXCurses.libs.handlers.filechooser.HandlersFileChooser, GLXCurses.
libs.handlers.label.HandlersLabel, GLXCurses.libs.handlers.statusbar.
HandlersStatusbar, GLXCurses.libs.handlers.textview.HandlersTextView,
GLXCurses.libs.handlers.widget.HandlersWidget, GLXCurses.libs.handlers.
window.HandlersWindow
```

### GLXCurses.libs.ChildElement module

```
class GLXCurses.libs.ChildElement.ChildElement (widget=None, widget_name=None, wid-
get_type=None, widget_id=None, wid-
get_properties=None)
    Bases: object
```

**widget**  
**name**  
**type**  
**id**  
**properties**

## GLXCurses.libs.ChildProperty module

```
class GLXCurses.libs.ChildProperty.ChildProperty (expand=None, fill=None,  
                                                    pack_type=None, padding=None,  
                                                    position=None)
```

Bases: object

### **expand**

Whether the child should receive extra space when the parent grows.

Note that the default value for this property is False for Box, but HBox, VBox and other subclasses use the old default of True.

Note: The “hexpand” or “vexpand” properties are the preferred way to influence whether the child receives extra space, by setting the child’s expand property corresponding to the box’s orientation.

In contrast to “hexpand”, the expand child property does not cause the box to expand itself.

Flags: Read / Write Default value: False

#### **Returns**

### **fill**

Whether the child should fill extra space or use it as padding.

Note: The “halign” or “valign” properties are the preferred way to influence whether the child fills available space, by setting the child’s align property corresponding to the box’s orientation to GLXC.ALIGN\_FILL to fill, or to something else to refrain from filling.

Flags: Read / Write

Default value: True

**Returns** If True the child fill extra space or use it as padding

**Return type** bool

### **pack\_type**

Whether the child should fill extra space or use it as padding.

Note: The “halign” or “valign” properties are the preferred way to influence whether the child fills available space, by setting the child’s align property corresponding to the box’s orientation to GLXC.ALIGN\_FILL to fill, or to something else to refrain from filling.

Flags: Read / Write

Default value: True

**Returns** If True the child fill extra space or use it as padding

**Return type** bool

**padding**

Extra space to put between the child and its neighbors, in chars.

Flags: Read / Write

Allowed values: <= G\_MAXINT

Default value: 0

**Returns** Extra space to put between the child and its neighbors, in chars.

**Return type** int

**position**

Extra space to put between the child and its neighbors, in chars.

Flags: Read / Write

Allowed values: <= G\_MAXINT

Default value: 0

**Returns** Extra space to put between the child and its neighbors, in chars.

**Return type** int

**GLXCurses.libs.Colorable module**

```
class GLXCurses.libs.Colorable.Colorable
```

Bases: object

**background\_color\_normal**

Get the background color

If set to None, return a default value

**Returns** the background color

**Return type** str or *None*

**background\_color\_prelight**

Get the background color or GLXCurses.Style if not set

**Returns** the background color

**Return type** str or *None*

**foreground\_color\_normal****foreground\_color\_prelight****color\_normal****color\_prelight****color\_insensitive****GLXCurses.libs.Colors module**

```
class GLXCurses.libs.Colors.Colors
```

Bases: object

**itu\_recommendation**

Get `itu_recommendation` property value

Where:

<https://en.wikipedia.org/wiki/ITU-R>

[https://en.wikipedia.org/wiki/Rec.\\_601](https://en.wikipedia.org/wiki/Rec._601)

[https://en.wikipedia.org/wiki/Rec.\\_709](https://en.wikipedia.org/wiki/Rec._709)

[https://en.wikipedia.org/wiki/Rec.\\_2100](https://en.wikipedia.org/wiki/Rec._2100)

Allowed Value: 'BT.601', 'BT.709', 'BT.2100' Default Value: 'BT.601'

**Returns** `itu_recommendation` property value

**Return type** `str`

**color\_detection\_value****static curses\_color** (*color*)

A “translation” function that converts standard-intensity CGA color numbers (0 to 7) to curses color numbers, using the curses constant names like `COLOR_BLUE` or `COLOR_RED`

**Parameters** `color` –

**Returns** `curses.COLOR`

**static curses\_color\_pair\_number** (*fg*, *bg*)

A function to set an integer bit pattern based on the classic color byte

**Parameters**

- **fg** (*int*) – Foreground color
- **bg** (*int*) – Background color

**curses\_color\_pairs\_init** ()

It function create all possible color pairs

**Returns**

**static strip\_hash** (*str\_rgb*)

Strip leading # if exists.

**Parameters** **str\_rgb** (*str*) – the str it contain a # or not

**Returns** a str without #

**Return type** `str`

**get\_luma\_component\_rgb** (*r*, *g*, *b*)**static rgb\_to\_ansi16** (*r*, *g*, *b*)**rgb\_to\_curses\_attributes** (*r*, *g*, *b*)**rgb\_hex\_to\_list\_int** (*str\_rgb*)**color** (*fg=None*, *bg=None*, *attributes=None*)

Convert a RGB value to a directly usable curses color

`draw(y, x, “Hello”, color)` where the return of it function is directly usable

**Returns** `color.pair` | `curses.Attribut`

**Return type** `int`

**hex\_rgb\_to\_curses** (*fg=None, bg=None*)  
 Convert a RGB value to a directly usable curses color  
 draw(y, x, "Hello", color) where the return of it function is directly usable  
 bg='#000000', FG='#FFFFFF'  
**Returns** color.pair | curses.Attribut  
**Return type** int

## GLXCurses.libs.Dividable module

**class** GLXCurses.libs.Dividable.**Dividable**  
 Bases: object

**static get\_child\_x\_coordinates** (*children=None, length=None*)  
 The function parse children list and calculate coordinates for each ChildElement by regarding ChildProperty information's and return a

**Parameters**

- **length** (*int*) – the max length in char
- **children** (*list of ChildElement*) – The list of children where we need coordinates

**Returns** a dict with child coordinates  
**Return type** dict of dict

**static get\_child\_y\_coordinates** (*children=None, length=None*)  
 The function parse children list and calculate coordinates for each ChildElement by regarding ChildProperty information's and return a

**Parameters**

- **length** (*int*) – the max length in char
- **children** (*list of ChildElement*) – The list of children where we need coordinates

**Returns** a dict with child coordinates  
**Return type** dict of dict

## GLXCurses.libs.File module

**class** GLXCurses.libs.File.**File**  
 Bases: object

**extension**  
**name**  
**directory**  
**path**  
**overwrite**  
**is\_binary()**

```
is_text()
found_best_output_file_name()
```

### GLXCurses.libs.FileChooserFunctions module

```
class GLXCurses.libs.FileChooserFunctions.FileChooserUtils
```

Bases: object

```
FILE_HIGH_LIGHT = {'archive': {'attributes': <MagicMock id='140675790644464'>, 'colo
```

```
FILE_HIGH_LIGHT_PREP = {'.3gp': {'attributes': <MagicMock id='140675790637952'>, 'ca
```

**directory\_view**

The directory view property is use to store the result of a scan directory.

**Returns** The view of the current directory as a list

**Return type** list of dict

**Raises** **TypeError** – When property value us not a list type or None

**cwd**

The cwd property store the location of the current directory value

If set to None it return os.getcwd() value

**Returns** The current working directory value

**Return type** str

**Raises**

- **TypeError** – When property value is not a str type or None
- **ValueError** – When property value is not a valid directory

**sort\_by\_name**

Return sort\_by\_name attribute.

**Returns** True if enable, False if disable

**Return type** bool

**sort\_name\_order**

Return sort\_name\_order attribute.

**Returns** True if ordering A to Z, False if ordering Z to A

**Return type** bool

**sort\_by\_size**

Return sort\_by\_size attribute.

**Returns** True if enable, False if disable

**Return type** bool

**category = 'database'**

**extension = 'ssql'**

**sort\_size\_order**

Return sort\_by\_size attribute. as set by set\_sort\_size\_order()

**Returns** True if enable, False if disable



**Return type** bool

**sort\_by\_mtime**

Return sort\_by\_mtime attribute.

**Returns** True if enable, False if disable

**Return type** bool

**sort\_mtime\_order**

Return sort\_mtime\_order attribute. as set by set\_sort\_mtime\_order()

**Returns** True if ordering Now to Ago, False if ordering Ago to Now.

**Return type** bool

**update\_directory\_list** ()

**set\_app\_file\_extensions** (*file\_extensions=None*)

A tuple of file extension to colorize, it's consider as file type you searching for.

The FileChooser will colorize they file's, in orange.

Note the function automatically deal with case sensitive.

Example: .mkv -> ('.mkv', '.Mkv', 'MKV')

**Parameters** **file\_extensions** (*tuple or None*) – a tuple of file extension to colorize or None for disable the colorize.

**Raises** **TypeError** – when file\_extensions argument is not a tuple type or None

**get\_app\_file\_extensions** ()

Return the list of file extension to colorize.

See . Filechooser.set\_app\_file\_extensions() for more details.

**Returns** a tuple of file extension to colorize or None if disable.

**Return type** tuple or *None*

**get\_color\_by\_filename** (*filename=None*)

**get\_attributes\_by\_filename** (*filename=None, key=None, default=None*)

**get\_infos\_by\_filename** (*filename=None, key=None, default=None*)

**convert\_file\_attribute** ()

## GLXCurses.libs.Group module

**class** GLXCurses.libs.Group.Group

Bases: object

**members**

**position**

Extra space to put between the child and its neighbors, in chars.

Flags: Read / Write

Allowed values: <= G\_MAXINT

Default value: 0

**Returns** Extra space to put between the child and its neighbors, in chars.

**Return type** int

**widget**

**is\_member** (*widget=None*)

**add** (*widget=None*)

Adds widget to the group .

Typically used for group widget's , by example RadioButton, MenuElement, GlobalFocus

For more complicated layout containers such as Box or Grid, this function will pick default packing parameters that may not be correct.

**Parameters** **widget** (*GLXCurses.Widget*) – a widget to be placed inside container

**Raises** **TypeError** – if widget is not a instance of GLXCurses.Widget

**remove** (*widget=None*)

**up** ()

**down** ()

### GLXCurses.libs.GroupElement module

**class** GLXCurses.libs.GroupElement.**GroupElement** (*widget=None*)

Bases: object

**widget**

### GLXCurses.libs.Groups module

**class** GLXCurses.libs.Groups.**Groups**

Bases: object

**groups**

**position**

Extra space to put between the child and its neighbors, in chars.

Flags: Read / Write

Allowed values: <= G\_MAXINT

Default value: 0

**Returns** Extra space to put between the child and its neighbors, in chars.

**Return type** int

**group**

**is\_group** (*group=None*)

**add\_group** (*group=None*)

Adds group to the GLXCurses.Application groups list .

Typically used to permit GLXCurses.Application to manage Widgets Groups

For more complicated layout containers such as Box or Grid, this function will pick default packing parameters that may not be correct.

**Parameters** **group** (*GLXCurses.Group*) – a widget to be placed inside container

**Raises `TypeError`** – if `group` is not a instance of `Group`

**`remove_group`** (*group=None*)

**`up`** ()

**`down`** ()

## GLXCurses.libs.ImageConvert module

**class** GLXCurses.libs.ImageConvert.**ImageConvert**

Bases: *GLXCurses.libs.File.File*

**data**

Get data property

**Returns** image data as a list

**Return type** list

**hsp\_debug**

Get hsp\_debug property

**Returns** image hsp\_debug as a list

**Return type** list

**width\_max**

Get the width\_max property value

**Returns** width\_max property value

**Return type** int or *None*

**width\_original**

Get the width\_original property value

**Returns** width\_original property value

**Return type** int or *None*

**height\_max**

Get the height\_max property value

**Returns** height\_max property value

**Return type** int or *None*

**height\_original**

Get the height\_original property value

it property is use when the widget discover image size

**Returns** height\_original property value

**Return type** int or *None*

**is\_resized**

Whether the image will be resized directly on the widget.

**Returns** True or False

**Return type** bool

**load\_image** (*path=None*)

### GLXCurses.libs.Movable module

```
class GLXCurses.libs.Movable.Movable
    Bases: object

    x_offset
        "x_offset for add offset value to x position of a GLXCurses.Area attach to a GLXCurses.Widget.

    y_offset
        "y_offset for add offset value to y position of a GLXCurses.Area attach to a GLXCurses.Widget.

    justify
        Return the Justify of the Button

        Justify:

        • LEFT

        • CENTER

        • RIGHT

        Returns str

    position_type
        Return the Position Type

        GLXCurses.GLXC.PositionType *GLXCurses.GLXC.POS_TOP *GLX-
        Curses.GLXC.POS_CENTER *GLXCurses.GLXC.POS_BOTTOM

        Returns the position_type property value

        Return type str

    check_justification()

    check_position()
```

### GLXCurses.libs.Spot module

```
class GLXCurses.libs.Spot.Spot
    Bases: GLXCurses.libs.Groups.Groups

    active_widgets

    active_window_id

    active_window_id_prev

    has_default

    has_focus

    has_prelight

    get_tooltip()
        Return the unique id of the widget it have been set by Application.set_tooltip()

        See also:

        Application.set_tooltip()

        Widget.id
```

**Returns** a unique id generate by uuid module

**Return type** long or *None*

**set\_tooltip** (*widget=None*)

Determines if the widget have to display a tooltip

“Not implemented yet”

**See also:**

`Application.get_tooltip()`

**Parameters** **widget** (*GLXCurses.Widget* or *None*) – a Widget

**permit\_keyboard\_interruption**

## GLXCurses.libs.TTY module

**class** `GLXCurses.libs.TTY.Singleton` (*name, bases, dictionary*)

Bases: type

**class** `GLXCurses.libs.TTY.Screen`

Bases: object

**stdscr**

**cbreak**

Normally, the tty driver buffers typed characters until a newline or carriage return is typed. If `cbreak=True` The cbreak property disables line buffering and erase/kill character-processing (interrupt and flow control characters are unaffected), making characters typed by the user immediately available to the program.

The (`cbreak` is `False`) returns the terminal to normal (cooked) mode.

Initially the terminal may or may not be in cbreak mode, as the mode is inherited; therefore, a program should call `cbreak=True` or `cbreak=False` explicitly.

Most interactive programs using curses set the `cbreak=True` mode.

Note that `cbreak` overrides `raw`.

The `raw=False` and `cbreak=False` calls follow historical practice in that they attempt to restore to normal ('cooked') mode from raw and cbreak modes respectively. M

Mixing (`raw` is `True` or `False`) and (`cbreak` is `True` or `False`) calls leads to tty driver control states that are hard to predict or understand; it is not recommended.

Note that return `None` if the property have never been set.

[See `curs_getch(3X)` for a discussion of how these routines interact with `echo=True` and `echo=False`.]

**Returns** The cbreak property value

**Return type** bool

**echo**

Control whether characters typed by the user are echoed by `getch` as they are typed.

Echoing by the tty driver is always disabled, but initially `getch` is in echo mode, so characters typed are echoed.

Authors of most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho`.

[See `curs_getch(3X)` for a discussion of how these routines interact with `cbreak` and `nocbreak`.]

:return the echo property value :rtype: bool

### **halfdelay**

The `halfdelay` property is used for half-delay mode, which is similar to `cbreak` mode in that characters typed by the user are immediately available to the program.

However, after blocking for `tenths` tenths of seconds, `ERR` is returned if nothing has been typed.

#### **Returns**

### **intrflush**

If the `intrflush` property is enabled, (`bf` is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing curses to have the wrong idea of what is on the screen.

Disabling (`bf` is `FALSE`), the option prevents the flush.

The default for the option is inherited from the tty driver settings. The window argument is ignored.

Note: That return `None` only if property have never been set

**Returns** The `intrflush` property value

**Return type** bool or *None*

### **keypad**

The `keypad` property enables the keypad of the user's terminal.

If enabled (`bf` is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (`bf` is `FALSE`), curses does not treat function keys specially and the program has to interpret the escape sequences itself.

If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch` is called.

The default value for `keypad` is `True`.

Note: That return `None` if `keypad` have never been set.

**Returns** The `keypad` property value

**Return type** bool or *None*

**instance** = `<GLXCurses.libs.TTY.Screen object>`

### **meta**

Initially, whether the terminal returns `def_prog_mode` 7 or 8 significant bits on input depends on the control mode of the tty driver [see `termio(7)`].

To force 8 bits to be returned, invoke `meta`` = ``True` this is equivalent, under POSIX, to setting the `CS8` flag on the terminal.

To force 7 bits to be returned, invoke `meta`` = ``False` this is equivalent, under POSIX, to setting the `CS7` flag on the terminal.

If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta`` = ``True` is called and `rmm` is sent when `meta`` = ``False` is called.

Note: That return `None` when the property have never been set

**Returns** The `meta` property value

**Return type** bool or *None*

### **nodelay**

The nodelay option causes getch to be a non-blocking call. If no input is ready, getch returns ERR. If disabled (bf is FALSE), getch waits until a key is pressed.

While interpreting an input escape sequence, wgetch sets a timer while waiting for the next character. If notimeout(win, TRUE) is called, then wgetch does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

**Returns** The nodelay property value

**Return type** bool or *None*

### **raw**

The raw property place the terminal into or out of raw mode.

Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal.

The behavior of the BREAK key depends on other bits in the tty driver that are not set by curses.

**Returns** The property value

**Return type** bool or *None*

### **qiflush**

When (qiflush is False) normal flush of input and output queues associated with the INTR, QUIT and SUSP characters will not be done [see termio(7)].

When (qiflush is True) is called, the queues will be flushed when these control characters are read.

You may want use (qiflush is False) in a signal handler if you want output to continue as though the interrupt had not occurred, after the handler exits.

**Returns** The qiflush property value

**Return type** bool or *None*

### **timeout**

The timeout and wtimeout routines set blocking or non-blocking read for a given window.

If delay is negative, blocking read is used (i.e., waits indefinitely for input).

If delay is zero, then non-blocking read is used (i.e., read returns ERR if no input is waiting).

If delay is positive, then read blocks for delay milliseconds, and returns ERR if there is still no input.

Hence, these routines provide the same functionality as nodelay, plus the additional capability of being able to block for only delay milliseconds (where delay is positive).

**Returns**

### **close()**

A Application must be close properly for permit to Curses to clean up everything and get back the tty in startup condition

Generally that is follow by a sys.exit(0) for generate a exit code.

### **lowlevel\_getch()**

Use by the Mainloop for interact with teh keyboard and the mouse.

getch() returns an integer corresponding to the key pressed.

If it is a normal character, the integer value will be equivalent to the character. Otherwise it returns a number which can be matched with the constants defined in `curses.h`.

For example if the user presses F1, the integer returned is 265.

This can be checked using the macro `KEY_F()` defined in `curses.h`.

This makes reading keys portable and easy to manage.

```
ch = GLXCurses.Screen().lowlevel_getch()
```

`lowlevel_getch()` will wait for the user to press a key, (unless you specified a timeout) and when user presses a key, the corresponding integer is returned.

Then you can check the value returned with the constants defined in `curses.h` to match against the keys you want.

```
if ch == curses.KEY_LEFT
    print("Left arrow is pressed")
```

**Returns** an integer corresponding to the key pressed.

**Return type** int

**reset\_screen()**

**refresh()**

**touch\_screen()**

**static check\_terminal** (*force\_xterm=False*)

**static get\_mouse()**

## **GLXCurses.libs.TextAttributes module**

**class** GLXCurses.libs.TextAttributes.**TextAttributes**

Bases: object

**attributes**

A list of style attributes to apply to the text of the label.

**Returns** A list of style attributes

**Return type** list

**label**

The contents of the label.

If the string contains TXT Markdown, you will have to set the `use_markdown` property to `True` in order for the label to display the Markdown attributes.

See also `set_markdown()` for a convenience function that sets both this property and the `use_markdown` property at the same time.

If the string contains underlines acting as mnemonics, you will have to set the `use_underline` property to `True` in order for the label to display them.

**Returns** The content of the label

**Return type** str



**markdown\_is\_used**

Get the `markdown_is_used` property value

Default value: False

**Returns** if True the contain of `label` property is consider as Markdown

**Return type** bool

**mnemonic\_char**

A string with `_` characters in positions correspond to characters in the text to underline.

**Returns** characters in the text use for underline

**Return type** str

**mnemonic\_is\_used**

Get the `mnemonic_is_used` property value

Default value: False

**Returns** if True the contain of `label` property is consider as Markdown

**Return type** bool

**mnemonic\_use\_underline**

If set, an underline in the text indicates the next character should be used for the mnemonic accelerator key.

Default value: False

**Returns** True if underline is display on text when use a mnemonic accelerator key

**Return type** bool

**new()**

Create a new `GLXCurses.TextAttributes` object and return it

**Returns** a new `GLXCurses.TextAttributes`

**Return type** `GLXCurses.TextAttributes`

**prepare\_attributes()****parse\_markdown\_with\_mnemonic()****parse\_markdown\_with\_no\_mnemonic()****parse\_text()**

**parse** (*label=None, markdown\_is\_used=None, mnemonic\_is\_used=None, mnemonic\_char=None, mnemonic\_use\_underline=None*)

**GLXCurses.libs.TextFonts module**

**class** `GLXCurses.libs.TextFonts.TextFonts`

Bases: `object`

**GLXCurses.libs.TextUtils module**

**class** `GLXCurses.libs.TextUtils.TextUtils`

Bases: `object`

**height**

Get *height* property value.

**Returns** *height* property

**Return type** int or *None*

**lines**

The lines This property has no effect if the text is not wrapping .

**Returns** Lines dispatch on by list item

**Return type** list

**text**

The contents of the label.

If the string contains TXT MarkDown, you will have to set the `use_markdown` property to True in order for the label to display the MarkDown attributes.

See also `set_markdown()` for a convenience function that sets both this property and the `use_markdown` property at the same time.

If the string contains underlines acting as mnemonics, you will have to set the `use_underline` property to True in order for the label to display them.

**Returns** The content of the label

**Return type** str

**width**

Get *width* property value.

**Returns** *width* property

**Return type** int or *None*

**wrap**

If set, wrap lines if the text becomes too wide.

**Returns** True if wrap is in use

**Return type** bool

**wrap\_mode**

If line wrapping is on (see the `wrap` property) this controls how the line wrapping is done.

The default is `GLXC.WRAP_WORD`, which means wrap on word boundaries.

**Returns** How the line wrapping is done

**Return type** `GLXC.WrapMode`

**scan()**

**text\_wrap** (*height=None, width=None*)

**GLXCurses.libs.Utils module**

`GLXCurses.libs.Utils.check_mnemonic_in_text` (*text=None, mnemonic\_char=None*)

`GLXCurses.libs.Utils.glxc_type` (*thing\_to\_test=None*)

Internal method for check if object pass as argument is GLXCurses Type Object

:param thing\_to\_test = A object to test :type thing\_to\_test: object :return: True or False :rtype: bool

`GLXCurses.libs.Utils.resize_text_wrap_char(text="", max_width=0)`

Resize the text , and return a new text

example: return '123' for '123456789' where max\_width = 3

**Parameters**

- **text** (*str*) – the original text to resize
- **max\_width** (*int*) – the size of the text

**Returns** a resize text

**Return type** str

`GLXCurses.libs.Utils.sizeof(value=None)`

Convert a num to a human readable thing it use metric prefix.

**Parameters** **value** (*int or float*) – a value to translate for a future display

**Returns** str

**Raises** **TypeError** – when value argument is not a int or float

`GLXCurses.libs.Utils.disk_usage(path)`

Return something like: 94G/458G (20%).

It use teh File system it self and just request it about the drive space where is store teh file pass in argument.

**Parameters** **path** –

**Type**

**Returns** something like 94G/458G (20%) with '.' as path argument

`GLXCurses.libs.Utils.round_up(n, decimals=0)`

<https://realpython.com/python-rounding/>

**Parameters**

- **n** –
- **decimals** –

**Returns**

`GLXCurses.libs.Utils.round_down(n, decimals=0)`

<https://realpython.com/python-rounding/>

**Parameters**

- **n** (*int or float*) – the number
- **decimals** (*int*) – number of decimal

**Returns** the rounded value

**Return type** int or float

`GLXCurses.libs.Utils.round_half_up(n, decimals=0)`

<https://realpython.com/python-rounding/>

**Parameters**

- **n** –
- **decimals** –

**Returns**

`GLXCurses.libs.Utils.round_half_down(n, decimals=0)`  
<https://realpython.com/python-rounding/>

**Parameters**

- **n** –
- **decimals** –

**Returns**

`GLXCurses.libs.Utils.resize_text(text="", max_width=0, separator='~')`  
Resize the text, and return a new text

example: return '123~789' for '123456789' where max\_width = 7 or 8

**Parameters**

- **text** (*str*) – the original text to resize
- **max\_width** (*int*) – the size of the text
- **separator** (*str*) – a separator a in middle of the resize text

**Returns** a resize text

**Return type** str

`GLXCurses.libs.Utils.clamp_to_zero(value=None)`  
Convert any int value to positive int

**Parameters** **value** (*int* or *None*) – a integer

**Returns** a integer

**Return type** int

`GLXCurses.libs.Utils.clamp(value=None, smallest=None, largest=None)`  
Back value inside smallest and largest value range.

**Parameters**

- **value** (*int* or *float*) – The value it have to be clamped
- **smallest** – The lower value
- **largest** – The upper value

**Returns** The clamped value it depend of parameters value type, int or float will be preserve.

**Return type** int or float

`GLXCurses.libs.Utils.new_id()`  
Generate a GLXCurses ID like 'E59E8457', two chars by two chars it's a random HEX

**Default size:** 8 **Default chars:** 'ABCDEF0123456789'

**Benchmark**

Iteration	Duration	CPU Information
10000000	99.114s	Intel(R) Core(TM) i7-2860QM CPU @ 2.50GHz
1000000	9.920s	Intel(R) Core(TM) i7-2860QM CPU @ 2.50GHz
100000	0.998s	Intel(R) Core(TM) i7-2860QM CPU @ 2.50GHz
10000	0.108s	Intel(R) Core(TM) i7-2860QM CPU @ 2.50GHz

**Returns** a string it represent a unique ID

**Return type** str

`GLXCurses.libs.Utils.is_valid_id(value)`

Check if it's a valid id

**Parameters** `value` – a id to verify

**Returns** bool

`GLXCurses.libs.Utils.merge_dicts(*dict_args)`

A merge dict fully compatible Python 2 and 3

Given any number of dicts, shallow copy and merge into a new dict, precedence goes to key value pairs in latter dicts.

`GLXCurses.libs.Utils.get_os_temporary_dir()`

Get the OS default dir , the better as it can.

It suppose to be cross platform

**Returns** A tmp dir path

**Return type** str

## GLXCurses.libs.XDGBaseDirectory module

`GLXCurses.libs.XDGBaseDirectory.control_directory(directory=None, mode=448)`

Internal function it create a directory if not exist

**Parameters**

- **directory** –
- **mode** (*int*) – the permission mode of the directory if created example:0o700

**Type** str

**Returns** the directory path or None

**Return type** str

**Raises**

- **TypeError** – when `directory` is not a str type
- **TypeError** – when `directory` is not a int type

**class** `GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory`

Bases: object

**xdg\_data\_home**

`$XDG_DATA_HOME` defines the base directory relative to which user specific data files should be stored.

If `$XDG_DATA_HOME` is either not set or empty, a default equal to `$HOME/.local/share` should be used.

**Returns** base directory relative to which user specific data files

**Return type** str

**xdg\_config\_home**

`$XDG_CONFIG_HOME` defines the base directory relative to which user specific configuration files should be stored.

If `$XDG_CONFIG_HOME` is either not set or empty, a default equal to `$HOME/.config` should be used.

**Returns** base directory relative to which user specific configuration files

**Return type** str

**xdg\_data\_dirs**

\$XDG\_DATA\_DIRS defines the preference-ordered set of base directories to search for data files in addition to the \$XDG\_DATA\_HOME base directory.

The directories in \$XDG\_DATA\_DIRS should be separated with a colon ‘:’.

If \$XDG\_DATA\_DIRS is either not set or empty, a value equal to /usr/local/share/:/usr/share/ should be used.

**Returns** preference-ordered set of base directories to search for data files separated with a colon ‘:’

**Return type** str

**xdg\_config\_dirs**

\$XDG\_CONFIG\_DIRS defines the preference-ordered set of base directories to search for configuration files in addition to the \$XDG\_CONFIG\_HOME base directory.

The directories in \$XDG\_CONFIG\_DIRS should be separated with a colon ‘:’.

If \$XDG\_CONFIG\_DIRS is either not set or empty, a value equal to /etc/xdg should be used.

**Returns** preference-ordered set of base directories to search for configuration files separated with a colon ‘:’

**Return type** str

**xdg\_cache\_home**

\$XDG\_CACHE\_HOME defines the base directory relative to which user specific non-essential data files should be stored.

If \$XDG\_CACHE\_HOME is either not set or empty, a default equal to \$HOME/.cache should be used.

**Returns** base directory relative to which user specific non-essential data files

**Return type** str

**xdg\_runtime\_dir**

\$XDG\_RUNTIME\_DIR defines the base directory relative to which user-specific non-essential runtime files and other file objects (such as sockets, named pipes, ...) should be stored.

If \$XDG\_RUNTIME\_DIR is not set applications should fall back to a replacement directory with similar capabilities and print a warning message.

Applications should use this directory for communication and synchronization purposes and should not place larger files in it, since it might reside in runtime memory and cannot necessarily be swapped out to disk.

**Returns** base directory relative to which user-specific non-essential runtime files

**Return type** str

**resource**

resource should normally be the name of your application or a shared resource.

**Returns** name of your application or a shared resource

**Return type** str

**set\_resource (\*resource)**

Set the resource property value.

**Parameters** `resource` (*tuple like 'Hello', '42'*) – should normally be the name of your application

**Raises** `AssertionError` – If when join the tuple start by `os.path.sep` typically /

#### **config\_path**

Ensure `$XDG_CONFIG_HOME/<resource>/` exists, and return its path.

**Returns** `$XDG_CONFIG_HOME/<resource>/` path

**Return type** str

#### **config\_paths**

Returns an iterator which gives each directory named 'resource' in the configuration search path.

Information provided by earlier directories should take precedence over later ones, and the user-specific config dir comes first.

**Returns** pre-ordered set of base directories to search for configuration files directory for resource

**Return type** list

#### **data\_path**

Ensure `$XDG_DATA_HOME/<resource>/` exists, and return its path.

**Returns** `$XDG_DATA_HOME/<resource>/` path

**Return type** str

#### **data\_paths**

Returns an iterator which gives each directory named 'resource' in the application data search path.

Information provided by earlier directories should take precedence over later ones.

**Returns** preference-ordered set of base directories to search for data files directory for resource

**Return type** list

#### **cache\_path**

Ensure `$XDG_CACHE_HOME/<resource>/` exists, and return its path.

**Returns** `$XDG_CACHE_HOME/<resource>/` path

**Return type** str

## **Module contents**

### **Submodules**

### **GLXCurses.Actionable module**

#### **class GLXCurses.Actionable.Actionable**

Bases: object

Actionable — An interface for widgets that can be associated with actions

#### **Known Implementations**

**Actionable is implemented by GLXC.Actionable and contain a list of widget** Button, CheckButton, CheckMenuItem, ColorButton, FontButton, ImageMenuItem, LinkButton, ListBoxRow, LockButton,

MenuButton, MenuItem, MenuToolButton, ModelButton, RadioButton, RadioMenuItem, RadioToolButton, ScaleButton, SeparatorMenuItem, Switch, TearoffMenuItem, ToggleButton, ToggleToolButton, ToolButton, VolumeButton.

**action\_name**

**action\_target**

**get\_action\_name()**

Gets the action name for *actionable*.

See `set_action_name()` for more information.

**Returns** the action name, or *None* if unset.

**Return type** *str* or *None*

**set\_action\_name** (*action\_name=None*)

Specifies the name of the action with which this widget should be associated. If *action\_name* is *NULL* then the widget will be unassociated from any previous action.

Usually this function is used when the widget is located (or will be located) within the hierarchy of a *ApplicationWindow*.

Names are of the form “win.save” or “app.quit” for actions on the containing *ApplicationWindow* or its associated *GLXCurses.Application*, respectively.

This is the same form used for actions in the *GMenu* associated with the window.

**Parameters** *action\_name* (*str* or *None*) – an action name, or *None*.

**Raises** **TypeError** – if *action\_name* is not a *str* type or *None*

**get\_action\_target\_value()**

Gets the current target value of *actionable*.

See `gtk_actionable_set_action_target_value()` for more information.

**Returns** the current target value.

**Return type** *GLXCurses.Object* or *None*

**set\_action\_target\_value** (*target\_value=None*)

Gets the current target value of *actionable*.

See `gtk_actionable_set_action_target_value()` for more information.

**Parameters** *target\_value* (*GLXCurses.Object* or *None*) – the target value, or *NULL*

## GLXCurses.Adjustment module

**class** *GLXCurses.Adjustment.Adjustment*

Bases: *GLXCurses.Object.Object*

A representation of an adjustable bounded value

### Properties

**lower**

The minimum value of the adjustment.

**Type** *float*

**Flags** *Read / Write*



**Default value** 0.0

**page\_increment**

The page increment of the adjustment.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**page\_size**

The page size of the adjustment. Note that the page-size is irrelevant and should be set to zero if the adjustment is used for a simple scalar value, e.g. in a `SpinButton`.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**step\_increment**

The step increment of the adjustment.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**minimum\_increment**

The smaller of step increment and page increment.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**upper**

The maximum value of the adjustment.

**Type** float

**Flags** Read / Write

**Default value** 0.0

---

**Note:** The values will be restricted by `upper - page-size` if the page-size property is nonzero.

---

**value**

The value of the adjustment.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**Description**

The *Adjustment* object represents a value which has an associated lower and upper bound, together with step and page increments, and a page size. It is used within several widgets, including `SpinButton`, `Viewport`, and *Range* (which is a base class for `Scrollbar` and `Scale`).

The Adjustment object does not update the value itself. Instead it is left up to the owner of the *Adjustment* to control the value.

### Functions

**new** (*value=0.0, lower=0.0, upper=0.0, step\_increment=0.0, page\_increment=0.0, page\_size=0.0*)  
Creates a new *GLXCurses.Adjustment*.

#### Parameters

- **value** (*float*) – The initial value
- **lower** (*float*) – The minimum value
- **upper** (*float*) – The maximum value
- **step\_increment** (*float*) – The step increment
- **page\_increment** (*float*) – The page increment
- **page\_size** (*float*) – The page size

**Returns** a new *GLXCurses.Adjustment*

**Return type** *GLXCurses.Adjustment*

#### Raises

- **TypeError** – if *value* is not float
- **TypeError** – if *lower* is not float
- **TypeError** – if *upper* is not float
- **TypeError** – if *step\_increment* is not float
- **TypeError** – if *page\_increment* is not float
- **TypeError** – if *page\_size* is not float

**get\_value** ()

Gets the current value of the adjustment. See *set\_value()*

**Returns** A current value Adjustment

**Return type** float

**set\_value** (*value*)

Set the *Adjustment value* attribute.

The value passed as argument is clamped to lie between *lower* and *lower* attributes.

---

**Note:** For adjustments which are used in a *Scrollbar*, the effective range of allowed values goes from *lower* to *upper - page\_size*.

---

**Raises** **TypeError** – when *value* passed as argument is not a *:py:\_\_area\_data:float*

**clamp\_page** (*lower=None, upper=None*)

Updates the *value* attribute to ensure that the range between *lower* and *upper* parameters is in the current page (i.e. between *value* and *value + page\_size*).

If the range is larger than the page size, then only the start of it will be in the current page. A **value-changed** signal will be emitted if the value is changed.

#### Parameters

- **lower** (*float*) – the lower value
- **upper** (*float*) – the upper value

**Raises**

- **TypeError** – when `lower` are not `:py:__area_data:float` type
- **TypeError** – when `upper` are not `:py:__area_data:float` type

**emit\_changed()**

Emits a “changed” signal from the *Adjustment*.

This is typically called by the owner of the *Adjustment*, after it has changed any of the *Adjustment* attributes other than the value.

**emit\_value\_changed()**

Emits a “value-changed” signal from the *Adjustment*. This is typically called by the owner of the *Adjustment* after it has changed the “value” property.

**configure** (*value=None, lower=None, upper=None, step\_increment=None, page\_increment=None, page\_size=None*)

Sets all properties of the adjustment at once.

Use this function to avoid multiple emissions of the “changed” signal.

See *Adjustment.set\_lower()* for an alternative way of compressing multiple emissions of “changed” into one.

**Parameters**

- **value** (*float*) – the new value
- **lower** (*float*) – the new minimum value
- **upper** (*float*) – the new maximum value
- **step\_increment** (*float*) – the new step increment
- **page\_increment** (*float*) – the new page increment
- **page\_size** (*float*) – the new page size

**Raises** **TypeError** – when one of parameters are not `:py:__area_data:float` type

**get\_lower()**

Retrieves the minimum value of the adjustment.

**Returns** The current minimum value of the adjustment

**Return type** float

**get\_page\_increment()**

Retrieves the page increment of the adjustment.

**Returns** The current page increment of the adjustment

**Return type** float

**get\_page\_size()**

Retrieves the page size of the adjustment.

**Returns** The current page size of the adjustment

**Return type** float

**get\_step\_increment()**

Retrieves the step increment of the adjustment.

**Returns** The current step increment of the adjustment.

**Return type** float

**get\_minimum\_increment()**

Get the smaller of step increment and page increment. Note that value is compute, then it have no need of a `set_minimum_increment()` method.

**Returns** the minimum increment of adjustment

**Return type** float

**get\_upper()**

Retrieves the maximum value of the adjustment.

**Returns** The current maximum value of the adjustment

**Return type** float

**set\_lower(lower)**

Sets the minimum value of the adjustment.

When setting multiple adjustment properties via their individual setters, multiple `Adjustment.changed()` signals will be emitted. However, since the emission of the `Adjustment.changed()` signal is tied to the emission of the `notify` signals of the changed properties, it's possible to compress the `Adjustment.changed()` signals into one by calling `object_freeze_notify()` and `object_thaw_notify()` around the calls to the individual setters.

Alternatively, using `Adjustment.configure()` has the same effect of compressing `Adjustment.changed()` emissions.

**Warning:** Unfortunately `object_freeze_notify()` and `object_thaw_notify()` don't exist yet. then only `Adjustment.configure()` will make the work.

**Parameters** `lower(float)` – the new minimum value

**Raises** **TypeError** – when “lower” argument is not a `:py:__area_data:float`

**set\_page\_increment(page\_increment)**

Sets the page increment of the adjustment.

**See also:**

`Adjustment.set_lower()` about how to compress multiple emissions of the `Adjustment.changed()` signal when setting multiple adjustment attributes.

**Parameters** `page_increment(float)` – the new page increment

**Raises** **TypeError** – when “page\_increment” argument is not a `:py:__area_data:float`

**set\_page\_size(page\_size)**

Sets the page size of the adjustment.

**See also:**

`Adjustment.set_lower()` about how to compress multiple emissions of the `Adjustment.changed()` signal when setting multiple adjustment attributes.

**Parameters** `page_size(float)` – the new page size

**Raises** **TypeError** – when “page\_size” argument is not a `:py:__area_data:float`

**set\_step\_increment** (*step\_increment*)  
Sets the step increment of the adjustment.

**See also:**

*Adjustment.set\_lower()* about how to compress multiple emissions of the *Adjustment.changed()* signal when setting multiple adjustment attributes.

**Parameters** *step\_increment* (*float*) – the new step increment

**Raises** **TypeError** – when “step\_increment” argument is not a `:py:__area_data:float`

**set\_upper** (*upper*)  
Sets the maximum value of the adjustment.

**See also:**

*Adjustment.set\_lower()* about how to compress multiple emissions of the *Adjustment.changed()* signal when setting multiple adjustment attributes.

**Parameters** *upper* (*float*) – the new maximum value

**Raises** **TypeError** – when “upper” argument is not a `:py:__area_data:float`

## GLXCurses.Aera module

**class** GLXCurses.Aera.**Area** (*x=None, y=None, width=None, height=None, screen=None, sub-win=None*)

Bases: object

Internal class it define a Area

---

**Note:** it never have a clamp value or a float to int conversion, each set method have role to raise a error if value type is not respect during a set.

---

**x**

× property

It represent the x location, 0 for Left

**Returns** × location in char, 0 correspond to left

**Return type** int or *None*

**y**

y location of the area

**Returns** y location in char, 0 correspond to top

**Return type** int

**width**

Get *width* property value.

**Returns** *width* property

**Return type** int or *None*

**height**

Get *height* property value.

Returns *height* property

Return type int or *None*

#### **stdscr**

Get the *stdscr* property value.

Returns A Curses window object

Return type *\_curses.curses* window or *None*

#### **subwin**

Get the *subwin* property value.

Returns A Curses window object

Return type *\_curses.curses* window or *None*

**add\_character** (*y=None, x=None, character=None, color=None*)

**insert\_character** (*y=None, x=None, character=None, color=None*)

**add\_string** (*y=None, x=None, text=None, color=None*)

**insert\_string** (*y=None, x=None, text=None, color=None*)

**add\_horizontal\_line** (*y=None, x=None, character=None, length=None, color=None*)

Display a horizontal line starting at (y, x) with length n consisting of the character character.

**add\_vertical\_line** (*y=None, x=None, character=None, length=None, color=None*)

Display a horizontal line starting at (y, x) with length n consisting of the character character.

**add\_rectangle** (*uly, ulx, lry, lrx*)

Draw a rectangle with corners at the provided upper-left and lower-right coordinates.

**draw\_background** (*color=None*)

## **GLXCurses.Application module**

**class** GLXCurses.Application.**Singleton** (*name, bases, dict*)

Bases: type

**class** GLXCurses.Application.**Application**

Bases: *glxeloop.bus.Bus*, *GLXCurses.Aera.Area*, *GLXCurses.libs.Spot.Spot*,  
*GLXCurses.libs.ApplicationHandlers.Handlers*

### **Description**

Create a Application singleton instance.

That class have the role of a Controller and a NCurses Wrapper.

It have particularity to not be a GLXCurses.Widget, then have a tonne of function for be a fake GLX-Curses.Widget.

From GLXCurses point of view everything start with it component. All widget will be display and store inside it component.

**get\_widget\_by\_id** (*widget\_id=None*)

### **active\_window**

Gets the “active\_window” for the application.

The active *Window* is the one that was most recently focused (within the application).

This window may not have the focus at the moment if another application has it — this is just the most recently-focused window within this application.

**Returns** the active *Window*, or *None* if there isn't one.

**Return type** *ChildElement* or *None*

#### **children**

Store the `children` property value

It property is use for store a stack of windows object use during choice of the active window

Default value: []

**Returns** `children` property value

**Return type** list

#### **app\_menu**

#### **menubar**

The `MenuModel` for the menubar.

**Returns** menubar property value

**Return type** `GLXCurses.MenuBar` or *None*

#### **register\_session**

#### **screensaver\_active**

#### **style**

The style of the Application, which contains information about how it will look (colors, etc).

The Application Style is impose to each widget

**Returns** a `GLXCurses.Style` instance

**Return type** `GLXCurses.Style`

**instance** = `<GLXCurses.Application.Application object>`

#### **statusbar**

#### **messagebar**

Sets the messagebar of application .

This can only be done in the primary instance of the application, after it has been registered. “startup” is a good place to call this.

**Returns** the messagebar property value

**Return type** `GLXCurses.MessageBar` or *None*

#### **toolbar**

#### **add\_window** (*window*)

Add a *Window* widget to the *Application* windows children's list.

This call can only happen after the application has started; typically, you should add new application windows in response to the emission of the “activate” signal.

This call is equivalent to setting the “application” property of window to application .

Normally, the connection between the application and the window will remain until the window is destroyed, but you can explicitly remove it with `application.remove_window()`.

Galaxie-Curses will keep the application running as long as it has any windows.

**Parameters** `window` (`GLXCurses.Window`) – a window to add

**Raises** **TypeError** – if window parameter is not a `Window` type

**remove\_window** (`window`)

Remove a `Window` widget from the `Application` windows children's list.

Set "application" and "parent" attribute of the `GLXCurses.Window` to `None`.

**Parameters** `window` (`GLXCurses.Window`) – a window to add

**Raises** **TypeError** – if window parameter is not a `Window` type

**get\_window\_by\_id** (`identifier=None`)

Returns the `GtkApplicationWindow` with the given ID.

**Parameters** `identifier` (`int`) – an identifier number

**Returns** the window with ID `identifier`, or `None` if there is no window with this ID.

**Return type** `int` or `None`

**Raises** **TypeError** – when `identifier` is not a `int` type

**refresh** ()

Refresh the `NCurses` Screen, and redraw each contain widget's

It's a central refresh point for the entire application.

**check\_sizes** ()

Just a internal method for compute every size.

It consist to a serial of testable function call

**get\_mouse** ()

**eveloop\_input\_event** ()

**eveloop\_cmd** ()

**eveloop\_finalization** ()

**eveloop\_dispatch\_application** (`detailed_signal`, `args`)

Flush Mainloop event to Child's father's for a Widget's recursive event dispatch

**Parameters**

- **detailed\_signal** (`str`) – a string containing the signal name
- **args** (`list`) – additional parameters `arg1`, `arg2`

**eveloop\_keyboard\_interruption** ()

## GLXCurses.Bin module

**class** `GLXCurses.Bin.Bin`

Bases: `GLXCurses.Container.Container`

A container with just one child

### Description

The `Bin` widget is a container with just one child. It is not very useful itself, but it is useful for deriving subclasses, since it provides common code needed for handling a single child widget.

Many `GLXCurses` widgets are subclasses of `Bin`, including



- *Window*
- *Button*
- *Frame*
- `HandleBox`
- `ScrolledWindow`

**get\_child()**

Gets the child of the `GLXCurses.Bin`, or `None` if the bin contains no child widget.

The returned widget does not have a reference added, so you do not need to unref it.

**Returns** the child of `GLXCurses.Bin` , or `None` if it does not have a child.

**Return type** `GLXCurses.Bin` or *None*

**GLXCurses.Bindings module****class** `GLXCurses.Bindings.Binding`

Bases: `object`

**Bindings**

Bindings — Key bindings for individual widgets

**Description:**

**add\_signal** (*binding\_set*, *keyval*, *modifiers*, *signal\_name*, *binding\_args=None*)

Override or install a new key binding for *keyval* with *modifiers* on *binding\_set* .

**Parameters**

- **binding\_set** – a `BindingSet` to add a signal to
- **keyval** – key value
- **modifiers** – key modifier
- **signal\_name** – signal name to be bound
- **binding\_args** (*list*) – list of `BindingArg` signal arguments.

**GLXCurses.Box module****class** `GLXCurses.Box.Box`

Bases: *GLXCurses.Container.Container*

**Description**

The *Box* widget organizes child widgets into a rectangular area.

**baseline\_position**

Gets the *baseline\_position* value.

**Returns** a `GLXC.BaselinePosition`

**Return type** `GLXC.BaselinePosition`

**homogeneous**

Returns whether the *Box* is homogeneous (all children's have the same size).

**See also:**

`Box.set_homogeneous()`

**Returns** `True` if the *Box* is homogeneous.

**Return type** `bool`

**spacing**

**new** (*orientation*=*'HORIZONTAL'*, *spacing*=*None*)

Creates a new *Box*.

**Parameters**

- **orientation** (*Orientation*) – the box's orientation. Default: `ORIENTATION_HORIZONTAL`
- **spacing** (*int* or *None*) – the number of characters to place by default between children. Default: 0

**Returns** a new *Box*.

**Raises**

- **TypeError** – if *orientation* is not `glxc.ORIENTATION_HORIZONTAL` or `glxc.ORIENTATION_VERTICAL`
- **TypeError** – if *spacing* is not `int` type or `None`

**pack\_start** (*child*=*None*, *expand*=*True*, *fill*=*True*, *padding*=*None*)

Adds child to *Box*, packed with reference to the start of *Box*.

**Parameters**

- **child** (*a GLXCurses Object*) – the widget to be added to *Box*
- **expand** (*bool*) – `True` if the new child is to be given extra space allocated to *Box* `<GLXCurses.Box.Box>`. The extra space will be divided evenly between all children that use this option
- **fill** (*bool*) – `True` if space given to child by the *expand* option is actually allocated to child, rather than just padding it. This parameter has no effect if *expand* is set to `False`. A child is always allocated the full height of a horizontal *Box* and the full width of a vertical *Box*. This option affects the other dimension.
- **padding** (*int* or *None*) – extra space in characters to put between this child and its neighbors, over and above the global amount specified by *spacing* attribute. If child is a widget at one of the reference ends of box, then padding pixels are also put between child and the reference edge of box

**Raises**

- **TypeError** – if *child* is not a `GLXCurses` type as tested by `glxc_type()`
- **TypeError** – if *expand* is not `bool` type
- **TypeError** – if *fill* is not `bool` type
- **TypeError** – if *padding* is not `int` or `None`

**pack\_end** (*child=None, expand=True, fill=True, padding=None*)  
 Adds child to GLXCurses.Box once to the end of GLXCurses.Box .

#### Parameters

- **child** (*GLXCurses.Widget*) – the widget to be added to GLXCurses.Box
- **expand** (*bool*) – True if the new child is to be given extra space allocated to GLXCurses.Box . The extra space will be divided evenly between all children that use this option
- **fill** (*bool*) – True if space given to child by the expand option is actually allocated to child, rather than just padding it. This parameter has no effect if expand is set to False. A child is always allocated the full height of a horizontal *Box* and the full width of a vertical *Box*. This option affects the other dimension.
- **padding** (*int or None*) – extra space in characters to put between this child and its neighbors, over and above the global amount specified by *spacing* attribute. If child is a widget at one of the reference ends of box , then padding pixels are also put between child and the reference edge of box

#### Raises

- **TypeError** – if child is not a instance of LXCurses.Widget
- **TypeError** – if expand is not bool type
- **TypeError** – if fill is not bool type
- **TypeError** – if padding is not int or None

**reorder\_child** (*child, position*)

Moves child to a new position in the list of *Box* children. The list contains widgets packed PACK\_START as well as widgets packed PACK\_END, in the order that these widgets were added to *Box*.

A widget's position in the *Box* children list determines where the widget is packed into *Box*. A child widget at some position in the list will be packed just after all other widgets of the same packing type that appear earlier in the list.

#### Parameters

- **child** (*Widget*) – the widget to move
- **position** (*int*) – the new position for child in the list of children of *Box*, starting from 0. If negative, indicates the end of the list.

#### Raises

- **TypeError** – if child is not a GLXCurses type as tested by `glxc_type()`
- **TypeError** – if position is not int type
- **TypeError** – if child is not a GLXCurses type as tested by `glxc_type()`

**query\_child\_packing** (*child*)

Obtains information about how child is packed into box or None if child is not found

**Return Key's:** widget: the Widget of the child to query expand: expand child property. fill: fill child property padding: padding child property. pack\_type: pack-type child property

**Parameters** **child** (*a Galaxie Widget*) – the Widget of to query

**Returns** information about how child is packed into box

**Return type** dict or *None*

**Raises `TypeError`** – if `child` is not a `GLXCurses` type as tested by `glxc_type()`

**`set_child_packing`** (*child, expand, fill, padding, pack\_type*)

Sets the way child is packed into box .

**Parameters**

- **`child`** (*Widget*) – the *Widget* of the child to set
- **`expand`** (*bool*) – the new value of the expand child property
- **`fill`** (*bool*) – the new value of the fill child property
- **`padding`** (*int*) – the new value of the padding child property
- **`pack_type`** (*PackType*) – the new value of the pack-type child property

**Raises**

- **`TypeError`** – if `child` is not bool type
- **`TypeError`** – if `expand` is not bool type
- **`TypeError`** – if `padding` is not int or None
- **`TypeError`** – if `pack_type` is not `glxc.PACK_START` or `glxc.PACK_END`

**`set_center_widget`** (*widget=None*)

Sets a center widget; that is a child widget that will be centered with respect to the full width of the box, even if the children at either side take up different amounts of space.

**Parameters** **`widget`** (*Widget* or None) – the *Widget* of the child to set

**Raises `TypeError`** – if `widget` is not a `GLXCurses` type as tested by `glxc_type()` or None

**`get_center_widget`** ()

Retrieves the center widget of the box.

**Returns** the center widget or None in case no center widget is set.

## GLXCurses.Button module

**class** `GLXCurses.Button.Button`

Bases: `GLXCurses.Widget.Widget`, `GLXCurses.libs.Movable.Movable`

**text**

**interface\_normal**

Get the `interface_normal` property value

It property is use for display a chars around the button when the widget has default

Default Value: “[ ]”

**Returns** `interface_normal` property value

**Return type** str

**interface\_selected**

Get the `interface_selected` property value

It property is use for display a chars around the button when the widget has default

Default Value: “[<>]”

**Returns** `interface_selected` property value

**Return type** `str`

**interface**

**color**

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**update\_preferred\_sizes()**

## GLXCurses.Buzzer module

`GLXCurses.Buzzer.play_sound(frequency, duration)`

Play a sound via the Buzzer of the computer

**Parameters**

- **frequency** (*int*) – frequency in Hertz (HZ) of the note to play
- **duration** (*int*) – how many time we play the note in Millisecond (ms)

**class** `GLXCurses.Buzzer.Buzzer`

Bases: `object`

**Description**

The famous buzzer class, why not implement a wireless protocol with the buzzer ?

**tempo**

Beats per minute (bpm) is a unit typically used as a measure of tempo

**Type** :`py: __area_data: float`

**Flags** Read / Write

**Default value** 110.0

**get\_tempo()**

Get the tempo attribute

**Returns** tempo attribute value is in BPM

**Return type** `float`

**set\_tempo(tempo=110.0)**

Set the tempo attribute

**Parameters** **tempo** (*float*) – tempo value in BPM

**get\_tempo\_to\_ms()**

Get actual tempo value in Millisecond (ms)

**Returns** tempo value in ms

**Return type** `int`

**get\_croche()**

Get the **Croche** it consist to devise the tempo by 2

**Returns** tempo value div by 2 in Millisecond (ms)

**Return type** `int`

**get\_double\_croche()**

Get the **Double Croche** it consist to devise the tempo by 4

**Returns** tempo value div by 4 in Millisecond (ms)

**Return type** int

**get\_triple\_croche()**

Get the **Triple Croche** it consist to devise the tempo by 8

**Returns** tempo value div by 8 in Millisecond (ms)

**Return type** int

**get\_blanche()**

Get the **Blanche** it consist to multiply the tempo by 2

**Returns** tempo value div by 2 in Millisecond (ms)

**Return type** int

**get\_triolet()**

Get the **Triolet** it consist to multiply the tempo by 3

**Returns** tempo value div by 3 in Millisecond (ms)

**Return type** int

**get\_notes()**

Get MIDI notes list , each item contain a list as container

**Notes Structure:** list(Octave, Midi\_Note\_Number, Note\_Name, Frequency\_Hz, Absolute\_Cents)

**Returns** the entry midi note list

**Return type** list(list(),list(),list())

**static get\_ms\_to\_tempo(ms)**

Get the conversion of a ms value to a tempo value

**Parameters** **ms** (*int*) – tempo value in Millisecond (ms)

**Returns** 60000 divided by Millisecond (ms) value

**Return type** float

**get\_tempo\_to\_hertz()**

Get the conversion of the tempo in BPM to the frequency in Hz

**Returns** tempo divided by 60

**Return type** int

**static get\_hertz\_to\_ms(hz)**

Get the conversion of a **Hz** value to a **ms** value

**Parameters** **hz** (*int*) – frequency in Hertz (**Hz**)

**Returns** the duration of the period frequency in ms (**ms**)

**Return type** int

## GLXCurses.CheckButton module

**class** GLXCurses.CheckButton.**CheckButton**

Bases: *GLXCurses.Widget.Widget, GLXCurses.libs.Movable.Movable*

```

active
text
interface
color
draw_widget_in_area ()
    Be here for be overwrite by every widget
update_preferred_sizes ()

```

## GLXCurses.Clipboards module

```
class GLXCurses.Clipboards.Clipboard
```

Bases: object

```
text
```

```
get ()
```

Returns the clipboard object for the given selection.

**Returns** The appropriate clipboard object.

**Return type** *GLXCurses.Clipboard*

```
set_text (clipboard=None, text=None, length=-1)
```

Sets the contents of the GLXCurses.Clipboard to the given UTF-8 string. GLXCurses will make a copy of the text and take responsibility for responding for requests for the text, and for converting the text into the requested format.

### Parameters

- **clipboard** (*GLXCurses.Clipboard* or *None*) – a GLXCurses.Clipboard object or None for self
- **text** (*str* or *None*) – a UTF-8 string.
- **length** (*int*) – length of text, in bytes, or -1, in which case the length will be determined with len().

```
wait_for_text (clipboard=None)
```

Requests the contents of the GLXCurses.Clipboard as text and converts the result to UTF-8 if necessary. This function waits for the `__area_data` to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

**Parameters** **clipboard** (*GLXCurses.Clipboard*) – a GLXCurses.Clipboard

**Returns** a newly-allocated UTF-8 string or NULL if retrieving the selection `__area_data` failed. This could happen for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into text form.).

**Return type** *str*

```
set_can_store (clipboard=None, targets=None, n_targets=None)
```

Hints that the clipboard `__area_data` should be stored somewhere when the application exits or when `store()` is called.

This value is reset when the clipboard owner changes.

### Parameters

- **clipboard** (`GLXCurses.Clipboard` or `None`) – a `GLXCurses.Clipboard` object or `None` for self
- **targets** (*TYPE Constant* or `None`) – array containing information about which forms should be stored or `None` to indicate that all forms should be stored.
- **n\_targets** (*int* or `None`) – number of elements in targets

**store** (*clipboard=None*)

Stores the current clipboard `__area_data` somewhere so that it will stay around after the application has quit.

**Parameters** **clipboard** (`GLXCurses.Clipboard` or `None`) – a `GLXCurses.Clipboard` object or `None` for self

## GLXCurses.Constants module

**class** `GLXCurses.Constants.Constants`

Bases: `object`

**exception** `ConstError`

Bases: `TypeError`

## GLXCurses.Container module

**class** `GLXCurses.Container.Container`

Bases: *GLXCurses.Widget.Widget*

`GLXCurses.Container` — Base class for widgets which contain other widgets

Description:

A `GLXCurses` user interface is constructed by nesting widgets inside widgets. Container widgets are the inner nodes in the resulting tree of widgets: they contain other widgets. So, for example, you might have a `GLXCurses.Window` containing a `GLXCurses.Frame` containing a `GLXCurses.Label`. If you wanted an image instead of a textual label inside the frame, you might replace the `GLXCurses.Label` widget with a `GLXCurses.Image` widget.

There are two major kinds of container widgets in `GLXCurses`. Both are subclasses of the abstract `GLXCurses.Container` base class.

The first type of container widget has a single child widget and derives from `GLXCurses.Bin`. These containers are decorators, which add some kind of functionality to the child. For example, a `GLXCurses.Button` makes its child into a clickable button; a `GLXCurses.Frame` draws a frame around its child and a `GLXCurses.Window` places its child widget inside a top-level window.

The second type of container can have more than one child; its purpose is to manage layout. This means that these containers assign sizes and positions to their children. For example, a `GLXCurses.HBox` arranges its children in a horizontal row, and a `GLXCurses.Grid` arranges the widgets it contains in a two-dimensional grid.

For implementations of `GLXCurses.Container` the virtual method `GLXCurses.Container.forall()` is always required, since it's used for drawing and other internal operations on the children. If the `GLXCurses.Container` implementation expect to have non internal children it's needed to implement both `GLXCurses.Container.add()` and `GLXCurses.Container.remove()`. If the `GLXCurses.Container` implementation has internal children, they should be added with `widget.set_parent()` on `__init__()` and removed with `widget.unparent()` in the `GLXCurses.Widget.destroy()` implementation. See more about implementing custom widgets at <https://wiki.gnome.org/HowDoI/CustomWidgets>



**border\_width**

Set the `border_width` property value

Allowed values:  $\leq 65535$

Default value: 0

**Returns** The width of the empty border outside the containers children.

**Return type** `int`

**child**

Set the `child` property value

**Returns** Child element

**Return type** `GLXCurses.ChildElement` or *None*

**resize\_mode**

Set the `resize_mode` property value

Default value: `GLXC.RESIZE_PARENT`

**Returns** Specify how resize events are handled.

**Return type** `str`

**add** (*widget=None*)

Adds widget to container .

Typically used for simple containers such as `Window`, `Frame`, or `Button`;

For more complicated layout containers such as `Box` or `Grid`, this function will pick default packing parameters that may not be correct.

So consider functions such as `GLXCurses.Box.pack_start()` and `GLXCurses.Grid.attach()` as an alternative to `GLXCurses.Container.add()` in those cases.

A widget may be added to only one container at a time; you (should not) place the same widget inside two different containers.

**Parameters** **widget** (*GLXCurses.Widget*) – a widget to be placed inside container

**Raises** **TypeError** – if *widget* is not a instance of `GLXCurses.Widget`

**remove** (*widget=None*)

Removes widget from container .

Widget must be inside container .

Note that container will own a reference to *widget* , and that this may be the last reference held; so removing a widget from its container can destroy that widget. If you want to use *widget* again, you need to add a reference to it before removing it from a container, using `g_object_ref()`. If you don't want to use *widget* again it's usually more efficient to simply destroy it directly using `Widget.destroy()` since this will remove it from the container and help break any circular reference count cycles.

**Parameters** **widget** (*GLXCurses Widget*) – a current child of container

**Raises** **TypeError** – if *widget* is not a instance of `GLXCurses.Widget`

**add\_with\_properties** (*widget=None, properties=None*)

Adds widget to container , setting child properties at the same time. See `GLXCurses.Container.add()` and `GLXCurses.Container.child_set()` for more details.

**Parameters**

- **widget** (*GLXCurses.Widget*) – a widget to be placed inside container

- **properties** (*GLXCurses.ChildProperty*) – properties to set

**Raises**

- **TypeError** – if `properties` is not a `GLXCurses.ChildProperty` instance
- **TypeError** – if `widget` is not a instance of `GLXCurses.Widget`

**get\_resize\_mode()**

Returns the resize mode for the container.

**Allowed value:**

- `GLXC.RESIZE_PARENT`
- `GLXC.RESIZE_QUEUE`
- `GLXC.RESIZE_IMMEDIATE`

**See also:**

`GLXCurses.Container.set_resize_mode()`.

**Warning:** `GLXCurses.Container.get_resize_mode()` has been deprecated since version 3.12 of GTK+, if will be remove as soon of possible.

**Returns** the current resize mode

**Return type** `GLXCurses.Constants`

**set\_resize\_mode(resize\_mode=None)**

Sets the resize mode for the container.

The resize mode of a container determines whether a resize request will be passed to the container's parent, queued for later execution or executed immediately.

**Allowed value:**

- `GLXC.RESIZE_PARENT`
- `GLXC.RESIZE_QUEUE`
- `GLXC.RESIZE_IMMEDIATE`

**See also:**

`GLXCurses.Container.get_resize_mode()`.

**Warning:** `GLXCurses.Container.set_resize_mode()` has been deprecated since version 3.12 of GTK+, if will be remove as soon of possible.

**Parameters** **resize\_mode** (*GLXCurses.Constants*) – the new resize mode

**check\_resize()**

The `check_resize()` method emits the “check-resize” signal on the container.

**foreachs(callback, \*callback\_data)**

Invokes callback on each non-internal child of container . See `GLXCurses.Container.forall()` for details on what constitutes an “internal” child. For all practical purposes, this function should iterate over precisely those child widgets that were added to the container by the application with explicit `add()` calls.

Most applications should use `GLXCurses.Container.foreachs()`, rather than `GLXCurses.Container.forall()`.

#### Parameters

- **callback** – a callback.
- **callback\_data** – callback user `__area_data`

**get\_path\_for\_child** (*child=None*)

Returns a newly created widget path representing all the widget hierarchy from the toplevel down to and including child .

**Returns** A newly created `WidgetPath`

**forall** (*callback, callback\_data*)

**propagate\_expose** (*child, event*)

**set\_focus\_chain** (*focusable\_widgets*)

**get\_focus\_chain** ()

**unset\_focus\_chain** ()

**set\_reallocate\_redraws** (*needs\_redraws*)

**set\_focus\_child** (*child*)

**get\_focus\_child** ()

**get\_focus\_vadjustment** ()

Retrieves the vertical focus adjustment for the container. See [Container.set\\_focus\\_vadjustment\(\)](#).

**Returns** the vertical focus adjustment, or `:py:__area_data:None` if none has been set.

**Return type** [Adjustment\(\)](#) or `:py:__area_data:None`

**set\_focus\_vadjustment** (*adjustment=None*)

Hooks up an adjustment to focus handling in a container, so when a child of the container is focused, the adjustment is scrolled to show that widget. This function sets the vertical alignment. See [scrolled\\_window\\_get\\_vadjustment\(\)](#) for a typical way of obtaining the adjustment and [Container.set\\_focus\\_hadjustment\(\)](#) for setting the horizontal adjustment.

The adjustments have to be in character units and in the same coordinate system as the allocation for immediate children of the container.

**Parameters** **adjustment** ([Adjustment\(\)](#) or `:py:__area_data:None`) – an adjustment which should be adjusted when the focus is moved among the descendants of container

**Raises** **TypeError** – if adjustment is not a [Adjustment\(\)](#)

**get\_focus\_hadjustment** ()

Retrieves the horizontal focus adjustment for the container. See [Container.set\\_focus\\_hadjustment\(\)](#).

**Returns** the horizontal focus adjustment, or `:py:__area_data:None` if none has been set.

**Return type** [Adjustment\(\)](#) or `:py:__area_data:None`

**set\_focus\_hadjustment** (*adjustment*)

Hooks up an adjustment to focus handling in a container, so when a child of the container is focused, the adjustment is scrolled to show that widget. This function sets the horizontal alignment. See [scrolled\\_window\\_get\\_hadjustment\(\)](#) for a typical way of obtaining the adjustment and [Container.set\\_focus\\_vadjustment\(\)](#) for setting the vertical adjustment.

The adjustments have to be in pixel units and in the same coordinate system as the allocation for immediate children of the container.

**Parameters** **adjustment** (*Adjustment ()* or `:py: __area_data:None`) – an adjustment which should be adjusted when the focus is moved among the descendants of `container`

**Raises** **TypeError** – if `adjustment` is not a *Adjustment ()*

**child\_type** (*container*)

Returns the type of the children supported by the container.

Note that this may return `None` to indicate that no more children can be added, e.g. for a `Paned` which already has two children.

Note that this may return `-1` to indicate `container` is not found

**Parameters** **container** –

**Returns** the type of children

**Return type** `str`, *None* or `-1`

**Raises** **TypeError** – if `child` is not a `GLXCurses` type as tested by `glxc_type ()`

**child\_set** (*child*, *properties=None*)

Sets one or more child properties for `child` and `container`.

**Parameters**

- **child** (*A GLXCurses.Widget*) – a `GLXCurses.Widget` which is a child of `container`
- **properties** (*GLXCurses.ChildProperty*) – properties to set

**Raises**

- **TypeError** – if `child` is not a `GLXCurses` type as tested by `glxc_type ()`
- **TypeError** – if `properties` is not a dict type

**child\_get** (*child*)

Gets the values of one or more child properties for `child` and `container`.

**Parameters** **child** (*A GLXCurses object*) – a widget which is a child of `container`

**Returns** properties of the child or `None` if child not found

**Return type** dict or *None*

**Raises** **TypeError** – if `child` is not a `GLXCurses` type as tested by `glxc_type ()`

**child\_set\_property** (*child*, *property\_name=None*, *value=None*)

Sets a child property for `child` and `container`.

**Parameters**

- **child** (*a GLXCurses.Widget*) – a `GLXCurses.Widget` which is a child of `GLXCurses.Container`
- **property\_name** (*str*) – the name of the property to set
- **value** (*everything except None*) – the value to set the property to

**Raises**

- **TypeError** – if `child` is not a `GLXCurses` type as tested by `glxc_type ()`
- **TypeError** – if `property_name` is not `str` type
- **TypeError** – if `value` is `None` type

**child\_get\_property** (*child*, *property\_name=None*)

Gets the value of a child property for child and container .

**Parameters**

- **child** (a *GLXCurses Object*) – a widget which is a child of container
- **property\_name** (*str*) – the name of the property to set

**Raises**

- **TypeError** – if *child* is not a GLXCurses type as tested by `glxc_type()`
- **TypeError** – if *property\_name* is not str type

**get\_border\_width** ()

Retrieves the border width of the container.

See GLXCurses.Container.set\_border\_width().

**Returns** the current border width

**Return type** int

**set\_border\_width** (*border\_width=0*)

Sets the border width of the container.

The border width of a container is the amount of space to leave around the outside of the container. The only exception to this is GLXCurses.Window; because toplevel windows can't leave space outside, they leave the space inside. The border is added on all sides of the container. To add space to only one side, use a specific "margin" property on the child widget, for example "margin-top".

border\_width have valid values are in the range 0-65535 chars and will be clamp to value.

**Parameters** **border\_width** (*int*) – amount of blank space to leave outside the container.

**Raises** **TypeError** – When border\_width is not a int

## GLXCurses.Dialog module

**class** GLXCurses.Dialog.Dialog

Bases: *GLXCurses.Window.Window*, *GLXCurses.libs.Movable.Movable*

**action\_aera\_border**

The default border width used around the action area of the dialog, as returned by Dialog.get\_action\_area(), unless GLXCurses.Container.set\_border\_width() was called on that widget directly.

**Returns** the action\_aera\_border property value

**Return type** int

**button\_spacing**

Spacing between buttons in Chars

**Returns** the button\_spacing property value

**Return type** int

**content\_area\_border**

The default border width used around the content area of the dialog, as returned by dialog\_get\_content\_area(), unless container\_set\_border\_width() was called on that widget directly.

**Returns** the content\_area\_border property value

**Return type** int

**content\_area\_spacing**

The default spacing used between elements of the content area of the dialog, as returned by `dialog_get_content_area()`, unless `box_set_spacing()` was called on that widget directly.

**Returns** the `content_area_spacing` property value

**Return type** `int`

**new\_with\_buttons** (*title, parent, \*flags*)**Parameters**

- **title** (*str* or `None`) – Title of the dialog, or `None`
- **parent** (*GLXCurses Parent* or `None`) – Transient parent of the dialog, or `None`
- **flags** (*argv*) –

**run** ()

Inform Application about the `GLXCurses.Dialog` is active.

Cause Application to forward event only inside the `GLXCurses.Dialog`.

The `GLXCurses.Mainloop` and `GLXCurses.Application` will work with the dialog like a normal `GLXCurses.Window` because dialog is a subclass of `GLXCurses.Window`.

**response** (*response\_id=None*)

Emits the “response” signal with the given response ID.

Used to indicate that the user has responded to the dialog in some way; typically either you or `Dialog().run()` will be monitoring the `::response` signal and take appropriate action.

**Parameters** **response\_id** (*str*) – response ID

**Raises** **TypeError** – when `response_id` is not a `str` type

**add\_button** (*button\_text=None, response\_id=None*)

Adds a button with the given `text` and sets things up so that clicking the button will emit the “response” signal with the given `response_id`.

The button is appended to the end of the dialog’s action area.

The button widget is returned, but usually you don’t need it.

**Parameters**

- **button\_text** (*str*) – text of button
- **response\_id** (*str*) – response ID for the button

**Returns** the `GLXCurses.Button` widget that was added.

**Return type** `GLXCurses.Button`

**Raises**

- **TypeError** – when `button_text` is not a `str` type
- **TypeError** – when `response_id` is not a `int` type

**add\_buttons** (*\*args*)

Adds more buttons, same as calling `Dialog.add_button()` repeatedly.

The data in arguments (`args`) must form a couple `button_text, response_id`.

Example: `Dialog.add_buttons('Hello.42', 42, 'Hello.43', 43, 'Hello.44', 44)`

Each button must have both text and response ID.

**Parameters** `args` – couple `button_text`, `response_id`

**add\_action\_widget** (`child=None`, `response_id=None`)

Adds an activatable widget to the action area of a `GLXCurses.Dialog`, connecting a signal handler that will emit the “response” signal on the dialog when the widget is activated.

The widget is appended to the end of the dialog’s action area.

If you want to add a non-activatable widget, simply pack it into the `action_area` field of the `GLXCurses.Dialog` struct.

**Parameters**

- **child** (`GLXCurses.Widget`) – an activatable widget
- **response\_id** (`str`) – response ID for child

**Raises**

- **TypeError** – when `child` is not a `GLXCurses.Widget` instance
- **TypeError** – when `response_id` is not a `str` type

**set\_default\_response** (`response_id=None`)

Sets the last widget in the dialog’s action area with the given `response_id` as the default widget for the dialog.

Pressing “Enter” normally activates the default widget.

**Parameters** `response_id` (`str`) – a response ID

**Raises** **TypeError** – when `response_id` is not a `str` type

**set\_response\_sensitive** (`response_id=None`, `setting=None`)

Calls `gtk_widget_set_sensitive` (`widget`, `@setting`) for each widget in the dialog’s action area with the given `response_id`.

A convenient way to sensitize/desensitize dialog buttons.

**Parameters**

- **response\_id** (`str`) – a response ID
- **setting** (`bool`) – True for sensitive

**Raises**

- **TypeError** – when `response_id` is not a `str` type
- **TypeError** – when `setting` is not a `bool` type

**get\_response\_for\_widget** (`widget=None`)

Gets the response id of a `GLXCurses.Widget` in the action area of a `GLXCurses.Dialog`.

Note: That the return `None` if the widget is not found in action area.

**Parameters** `widget` (`GLXCurses.Widget`) – a widget in the action area of dialog

**Returns** the response id of `GLXCurses.Widget`, or `GLXC.RESPONSE_NONE` if doesn’t have a response id.

**Return type** `int` or `GLXC.RESPONSE_NONE` or *None*

**get\_widget\_for\_response** (`response_id=None`)

Gets the widget button that uses the given response ID in the action area of a dialog.

**Parameters** `response_id` (`str`) – the response ID used by the dialog widget

**Returns** the widget button that uses the given `response_id`, or `None`.

**get\_action\_area()**

has been deprecated since version GTK3.12, GLXCurses return the internal `_action_area`.

Here the structure:

```
““ [
    { 'widget': button_widget, 'response_id': response_id, 'default_response': False
    }, {
        'widget': button_widget, 'response_id': response_id, 'default_response': False
    }
]
```

Returns the action area of dialog .

**Returns** the action area.

**Return type** list

**get\_content\_area()**

Returns the content area of dialog .

**Returns** the content area GLXCurses Box.

**Return type** GLXCurses.VBox

**close()**

Signal emitted when the user uses a keybinding to close the dialog.

**update\_preferred\_sizes()**

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

## GLXCurses.Editable module

**class** GLXCurses.Editable.Editable

Bases: object

**select\_region** (*editable=None, start\_pos=None, end\_pos=None*)

Selects a region of text. The characters that are selected are those characters at positions from `start_pos` up to, but not including `end_pos` . If `end_pos` is negative, then the characters selected are those characters from `start_pos` to the end of the text.

Note that positions are specified in characters, not bytes.

### Parameters

- **editable** (*GLXCurses.Editable or None*) – a GLXCurses.Editable
- **start\_pos** (*int or None*) – start of region
- **end\_pos** (*int or None*) – end of region

### Raises

- **TypeError** – if `start_pos` is not a int type or `None`.
- **TypeError** – if `end_pos` is not a int type or `None`.
- **TypeError** – if `editable` is not a valid GLXCurses type.



- **TypeError** – if `editable` is not a instance of `GLXCurses.Editable`.

**get\_selection\_bounds** (*editable=None*)

Retrieves the selection bound of the editable. `start_pos` will be filled with the start of the selection and `end_pos` with end. If no text was selected both will be identical and `FALSE` will be returned.

Note that positions are specified in characters, not bytes.

**Parameters** `editable` (*GLXC.Editable or None*) – a `GLXC.Editable`

**Returns** True if an area is selected, False otherwise

**Return type** bool

**Raises**

- **TypeError** – if `editable` is not a valid `GLXCurses` type.
- **TypeError** – if `editable` is not a instance of `GLXCurses.Editable`.

**insert\_text** (*editable=None, new\_text=None, new\_text\_length=-1, position=None*)

Inserts `new_text_length` bytes of `new_text` into the contents of the widget, at position `position`.

Note that the position is in characters, not in bytes.

The function updates position to point after the newly inserted text.

**Parameters**

- **editable** (*GLXC.Editable or None*) – a `GLXC.Editable`
- **new\_text** (*str*) – the text to append
- **new\_text\_length** (*int*) – the length of the text in bytes, or -1
- **position** (*int or None*) – location of the position text will be inserted at. None for insert at actual position.

**Raises**

- **TypeError** – if `editable` is not a valid `GLXCurses` type.
- **TypeError** – if `editable` is not a instance of `GLXCurses.Editable`.
- **TypeError** – if `new_text` is not a str or None.
- **TypeError** – if `new_text_length` is not a int or None.
- **TypeError** – if `position` is not a int or None.

**delete\_text** (*editable=None, start\_pos=None, end\_pos=None*)

Deletes a sequence of characters. The characters that are deleted are those characters at positions from `start_pos` up to, but not including `end_pos`.

If `end_pos` is negative, then the characters deleted are those from `start_pos` to the end of the text.

**Parameters**

- **editable** (*GLXC.Editable or None*) – a `GLXC.Editable`
- **start\_pos** (*int or None*) – start position
- **end\_pos** (*int or None*) – end position

**Raises**

- **TypeError** – if `editable` is not a valid `GLXCurses` type.
- **TypeError** – if `editable` is not a instance of `GLXCurses.Editable`.

- **TypeError** – if `start_pos` is not a int type or None.
- **TypeError** – if `end_pos` is not a int type or None.

**get\_chars** (*editable=None, start\_pos=None, end\_pos=None*)

Retrieves a sequence of characters. The characters that are retrieved are those characters at positions from `start_pos` up to, but not including `end_pos`.

If `end_pos` is negative, then the characters retrieved are those characters from `start_pos` to the end of the text.

Note that positions are specified in characters, not bytes.

#### Parameters

- **editable** (*GLXC.Editable or None*) – a GLXC.Editable
- **start\_pos** (*int*) – start of text
- **end\_pos** (*int*) – end of text

**Returns** a pointer to the contents of the widget as a string. This string is allocated by the GLXC.Editable implementation and should be freed by the caller.

#### Raises

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **ImportError** – if `editable` is not a instance of GLXCurses.Editable.
- **TypeError** – if `start_pos` is not a int type or None.
- **TypeError** – if `end_pos` is not a int type or None.

**cut\_clipboard** (*editable=None*)

Removes the contents of the currently selected content in the editable and puts it on the clipboard.

**Parameters** **editable** (*GLXCurses.Editable or None*) – a instance of GLXCurses.Editable.

#### Raises

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.

**copy\_clipboard** (*editable=None*)

Copies the contents of the currently selected content in the editable and puts it on the clipboard.

**Parameters** **editable** (*GLXCurses.Editable or None*) – a GLXCurses.Editable

#### Raises

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of Editable.

**paste\_clipboard** (*editable=None*)

Pastes the content of the clipboard to the current position of the cursor in the editable.

**Parameters** **editable** (*GLXC.Editable or None*) – a GLXC.Editable

#### Raises

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.

**delete\_selection** (*editable=None*)

Deletes the currently selected text of the editable. This call doesn't do anything if there is no selected text.

**Parameters** **editable** (*GLXC.Editable*) – a Class Name contain on the list GLXC.Editable

**Raises**

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.

**set\_position** (*editable=None, position=-1*)

Sets the cursor position in the editable to the given value.

The cursor is displayed before the character with the given (base 0) index in the contents of the editable. The value must be less than or equal to the number of characters in the editable.

A value of -1 indicates that the position should be set after the last character of the editable.

Note that position is in characters, not in bytes.

**Parameters**

- **editable** (*GLXC.Editable*) – a Class Name contain on the list GLXC.Editable
- **position** (*int*) – the position of the cursor

**Raises**

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.
- **TypeError** – if `position` is not a int type.

**get\_position** (*editable=None*)

Retrieves the current position of the cursor relative to the start of the content of the editable.

Note that this position is in characters, not in bytes.

**Parameters** **editable** (*GLXC.Editable*) – a Class Name contain on the list GLXC.Editable

**Returns** the cursor position

**Return type** int

**Raises**

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.

**set\_editable** (*editable=None, is\_editable=True*)

Determines if the user can edit the text in the editable widget or not.

**Parameters**

- **editable** (*GLXC.Editable*) – a Class Name contain on the list GLXC.Editable
- **is\_editable** (*bool*) – True if the user is allowed to edit the text in the widget

**Raises**

- **TypeError** – if `is_editable` is not a int type.
- **TypeError** – if `editable` is not a valid GLXCurses type.

- **TypeError** – if `editable` is not a instance of `GLXCurses.Editable`.

**get\_editable** (*editable=None*)

Retrieves whether `editable` is `editable`.

See `GLXCurses.Editable.set_editable()`.

**Parameters** `editable` (*GLXC.Editable*) – a Class Name contain on the list `GLXC.Editable`

**Returns** True if `editable` is `editable`.

**Raises**

- **TypeError** – if `editable` is not a valid `GLXCurses` type.
- **TypeError** – if `editable` is not a instance of `GLXCurses.Editable`.

## GLXCurses.Entry module

**class** `GLXCurses.Entry.Entry`

Bases: `GLXCurses.Widget.Widget`, `GLXCurses.Editable.Editable`, `GLXCurses.libs.Movable.Movable`

Entry — A single line text entry field

### Description

The `GLXCurses.Entry` widget is a single line text entry widget. A fairly large set of key bindings are supported by default. If the entered text is longer than the allocation of the widget, the widget will scroll so that the cursor position is visible.

When using an entry for passwords and other sensitive information, it can be put into “password mode” using `GLXCurses.Entry.set_visibility()`. In this mode, entered text is displayed using a “invisible” character. By default, `GLXCurses` picks the best invisible character that is available in the current font, but it can be changed with `GLXCurses.Entry.set_invisible_char()`. `GLXCurses` displays a warning when Caps Lock or input methods might interfere with entering text in a password entry. The warning can be turned off with the “caps-lock-warning” property.

**activates\_default**

Whether to activate the default widget (such as the default button in a dialog) when Enter is pressed.

Default Value: False

**Returns** True if the widget is the default widget

**Return type** bool

**Raises** **TypeError** – When `activates_default` property value is not a bool type or None

**attributes**

A list of attributes to apply to the text of the entry.

This is mainly useful to change the size or weight of the text.

The Attribute’s `start_index` and `end_index` must refer to the `EntryBuffer` text, i.e. without the `preedit` string.

Default Value: []

**Returns** attributes to apply to the text of the entry

**Return type** list

**Raises** **TypeError** – When `attributes` property value is not a list type or None

**buffer**

Text buffer object which actually stores entry text.

Default Value: `GLXCurses.EntryBuffer()`

**Returns** The Text buffer object

**Return type** `GLXCurses.EntryBuffer`

**Raises `TypeError`** – When `buffer` property value is not a `GLXCurses.EntryBuffer` instance or `None`

**caps\_locks\_warning**

Whether password entries will show a warning when Caps Lock is on.

Note that the warning is shown using a secondary icon, and thus does not work if you are using the secondary icon position for some other purpose.

Default Value: `True`

**Returns** `True` if a warning icon is display

**Return type** `bool`

**Raises `TypeError`** – When `caps_locks_warning` property value is not a `bool` type or `None`

**completion**

The auxiliary completion object to use with the entry.

**Returns** The Completion object

**Return type** `GLXCurses.EntryCompletion`

**Raises `TypeError`** – When `completion` property value is not a `GLXCurses.EntryCompletion` instance or `None`

**cursor\_position**

The current position of the insertion cursor in chars.

- Allowed values: `[0,65535]`
- Default Value: `0`

**Returns** `True` if the widget is the default widget

**Return type** `bool`

**Raises `TypeError`** – When `cursor_position` property value is not a `int` type or `None`

**editable**

Whether the entry contents can be edited.

**Returns** `True` if the entry contents can be edited

**Return type** `bool`

**Raises `TypeError`** – When `editable` property value is not a `bool` type or `None`

**enable\_emoji\_completion**

Whether to suggest Emoji replacements.

Default value: `False`

**Returns** `True` if the completion of emoji is enable

**Return type** `bool`

**Raises `TypeError`** – When `completion` property value is not a bool type or `None`

#### **`has_frame`**

FALSE removes outside bevel from entry.

Default value: `True`

**Returns** `False` removes outside bevel from entry.

**Return type** bool

**Raises `TypeError`** – When `has_frame` property value is not a bool type or `None`

#### **`im_module`**

Which IM (input method) module should be used for this entry. See `GLXCurses.IMContext`.

Setting this to a non-NULL value overrides the system-wide IM module setting. See the `GLXCurses.Settings` “`glxcurses_im_module`” property.

Default value: `None`

**Returns** IM (input method) as text

**Return type** str or *None*

**Raises `TypeError`** – When `im_module` property value is not a str type or `None`

#### **`inner_border`**

Sets the text area’s border between the text and the frame.

**Returns** a `GLXCurses.GLXC.BorderStyle` item

**Return type** str

**Raises**

- **`TypeError`** – When `inner_border` property value is not a str type or `None`
- **`Value`** – When `inner_border` property value is not a `GLXCurses.GLXC.BorderStyle`

#### **`input_hints`**

Additional hints (beyond “`input_purpose`”) that allow input methods to fine-tune their behaviour.

**Returns** `GLXC.InputHints` item

**Return type** str

**Raises**

- **`TypeError`** – When `input_hints` property value is not a str type or `None`
- **`Value`** – When `input_hints` property value is not a `GLXCurses.GLXC.InputHints`

#### **`input_purpose`**

The purpose of this text field.

This property can be used by on-screen keyboards and other input methods to adjust their behaviour.

Note that setting the purpose to `GLXC.INPUT_PURPOSE_PASSWORD` or `GLXC.INPUT_PURPOSE_PIN` is independent from setting “`visibility`”.

**Returns** a `GLXCurses.GLXC.InputPurpose` item

**Return type** str

**Raises**

- **`TypeError`** – When `input_purpose` property value is not a str type or `None`

- **Value** – When `input_purpose` property value is not a `GLXCurses.GLXC.InputPurpose` item

**invisible\_char**

The invisible character is used when masking entry contents (in “password mode”). When it is not explicitly set with the “invisible-char” property, `GLXCurses` determines the character to use from a list of possible candidates, depending on availability in the current font.

This style property allows the theme to prepend a character to the list of candidates.

Default value: `*`

**Returns** the invisible character

**Return type** `str`

**Raises `TypeError`** – When `invisible_char` property value is not a `str` type or `None`

**invisible\_char\_set**

Whether the invisible char has been set for the `GLXCurses.Entry`.

Default value: `False`

**Returns** `True` if char has been set for the `GLXCurses.Entry`.

**Return type** `bool`

**Raises `TypeError`** – When `invisible_char_set` property value is not a `bool` type or `None`

**max\_length**

Maximum number of characters for this entry. Zero if no maximum.

Default value: `0`

**Returns** Maximum number of characters for this entry

**Return type** `int`

**Raises `TypeError`** – When `max_length` property value is not a `bool` type or `None`

**max\_width\_chars**

The desired maximum width of the entry, in characters. If this property is set to `-1`, the width will be calculated automatically.

Allowed values: `>= -1`

Default value: `-1`

**Returns** The desired maximum width of the entry

**Return type** `int`

**Raises `TypeError`** – When `max_width_chars` property value is not a `int` type or `None`

**overwrite\_mode**

If text is overwritten when typing in the `GLXCurses.Entry`.

Default value: `False`

**Returns** `True` If text is overwritten when typing

**Return type** `bool`

**Raises `TypeError`** – When `overwrite_mode` property value is not a `bool` type or `None`

#### **placeholder\_text**

The text that will be displayed in the GtkEntry when it is empty and unfocused.

Default value: `None`

**Returns** IM (input method) as text

**Return type** str or *None*

**Raises** **TypeError** – When `im_module` property value is not a str type or `None`

#### **populate\_all**

If `populate_all` is `True`, the **populate-popup** signal is also emitted for touch popups.

Default value: `False`

**Returns** `True` If **populate-popup** signal is also emitted for touch popups.

**Return type** bool

**Raises** **TypeError** – When `populate_all` property value is not a bool type or `None`

#### **primary\_icon\_activatable**

Whether the primary icon is activatable.

GLXCurses emits the “icon-press” and “icon-release” signals only on sensitive, activatable icons.

Sensitive, but non-activatable icons can be used for purely informational purposes.

Default value: `True`

**Returns** `True` If emit “icon-press” and “icon-release” on sensitive, activable icon.

**Return type** bool

**Raises** **TypeError** – When `primary_icon_activatable` property value is not a bool type or `None`

#### **primary\_icon\_gicon**

The GIcon to use for the primary icon for the entry.

Default value: `None`

**Returns** something like a icon name

**Return type** str

**Raises** **TypeError** – When `primary_icon_gicon` property value is not a str type or `None`

#### **primary\_icon\_name**

The icon name to use for the primary icon for the entry.

Default value: `None`

**Returns** something like a icon name

**Return type** str

**Raises** **TypeError** – When `primary_icon_gicon` property value is not a str type or `None`

#### **primary\_icon\_sensitive**

Whether the primary icon is sensitive.

An insensitive icon appears grayed out. GLXCurses does not emit the “icon-press” and “icon-release” signals and does not allow DND from insensitive icons.

An icon should be set insensitive if the action that would trigger when clicked is currently not available.

Default value: `True`



**Returns** If `True` an insensitive icon appears grayed out.

**Return type** `bool`

**Raises** **TypeError** – When `primary_icon_sensitive` property value is not a `bool` type or `None`

#### **primary\_icon\_tooltip\_markdown**

The contents of the tooltip on the primary icon, which is marked up with the text markdown language

**Returns** The contents of the tooltip with markdown format

**Return type** `str` or *None*

#### **primary\_icon\_tooltip\_text**

The contents of the tooltip on the primary icon, which is marked up with the text format

**Returns** The contents of the tooltip with text format

**Return type** `str` or *None*

#### **progress\_fraction**

The current fraction of the task that's been completed.

Allowed values: `[0,1]`

Default value: `0`

**Returns** The the fraction value

**Return type** `float`

#### **draw\_widget\_in\_area()**

Be here for be overwrite by every widget

#### **new()**

Creates a new entry.

**Returns** A new `GLXCurses.Entry.Widget`

**Return type** `GLXCurses.Widget`

#### **new\_with\_buffer(buffer=None)**

Creates a new entry with the specified text buffer.

---

**Note:** `Utils.is_valid_id()` and `Utils.new_id()` are used for identify if the `buffer` is a Galaxie-Curses component. That `GLXCurses` ID is automatically generate at the widget creation.

---

**Parameters** **buffer** – The buffer to use for the new `GLXCurses.Entry`.

**Returns** A `Entry Buffer` object.

**Return type** `GLXCurses.Entry`

**Raises**

- **TypeError** – if `buffer` is not `GLXCurses.EntryBuffer` Type
- **TypeError** – if `buffer` haven't a valid `GLXCurses` ID

#### **get\_buffer()**

Get the `GLXCurses.EntryBuffer` object which holds the text for this widget.

**Returns** A `EntryBuffer` object.

**Return type** GLXCurse.Widget

**set\_buffer** (*buffer=None*)

Set the EntryBuffer object which holds the text for this widget.

**Parameters** **buffer** – The buffer to use for the GLXCurses.Entry.

**set\_text** (*text*)

Sets the text in the widget to the given value, replacing the current contents.

**See also:**

GLXCurses.EntryBuffer().set\_text()

**Parameters** **text** (*String*) – The new text

**get\_text** ()

Retrieves the contents of the entry widget. See also GLXCurses.Editable.get\_chars().

This is equivalent to: “ self.buffer = GLXCurses.EntryBuffer() self.buffer.get\_text() “ :return: A pointer to the contents of the widget as a string. This string points to internally allocated storage in the widget and must not be freed, modified or stored. :rtype: String

**get\_text\_length** ()

Retrieves the current length of the text in entry .

This is equivalent to: “ self.buffer = GLXCurses.EntryBuffer() self.buffer.get\_length() “

**Returns** The current number of characters in GtkEntry, or 0 if there are none.

**Return type** Int in range 0-65536

**set\_visibility** (*visible=None*)

Sets whether the contents of the entry are visible or not. When visibility is set to FALSE, characters are displayed as the invisible char, and will also appear that way when the text in the entry widget is copied elsewhere.

By default, GLXCurse picks the best invisible character available in the current font, but it can be changed with set\_invisible\_char().

---

**Note:** You probably want to set “input\_purpose” to glx.INPUT\_PURPOSE\_PASSWORD or glx.INPUT\_PURPOSE\_PIN to inform input methods about the purpose of this entry, in addition to setting visibility to FALSE.

---

**Parameters** **visible** (*bool*) – True if the contents of the entry are displayed as plaintext

**Raises** **TypeError** – if *visible* is not boolean type

**set\_invisible\_char** (*ch='\*'*)

Sets the character to use in place of the actual text when set\_visibility() has been called to set text visibility to FALSE.

---

**Note:** this is the character used in “password mode” to show the user how many characters have been typed.

---

By default, GLXCurse picks the best invisible char available in the current font.

---

**Note:** If you set the invisible char to 0, then the user will get no feedback at all; there will be no text on the stdscr as they type

---

**Parameters** `ch` (*str*) – a character

**Raises** **TypeError** – if `ch` is not printable str

**unset\_invisible\_char** ()

” Unset the invisible char previously set with `set_invisible_char()`. So that the default invisible char is used again.

**set\_max\_length** (*max=None*)

Sets the maximum allowed length of the contents of the widget. If the current contents are longer than the given length, then they will be truncated to fit.

**This is equivalent to:** `self.buffer = GLXCurses.EntryBuffer() self.buffer.set_max_length()`

**Parameters** `max` (*int*) – The maximum length of the entry, or 0 for no maximum. (other than the maximum length of entries.) The value passed in will be clamped to the range 0-65536.

**get\_activates\_default** ()

Retrieves the value set by `set_activates_default()`.

**Returns** TRUE if the entry will activate the default widget

**Return type** bool

**get\_has\_frame** ()

Gets the value set by `set_has_frame()`.

**Returns** whether the entry has a beveled frame

**Return type** bool

**get\_inner\_border** ()

This function returns the entry’s “inner-border” property. See `set_inner_border()` for more information.

GLXC.BorderStyle Members:

GLXC.BORDER\_STYLE\_NONE No visible border GLXC.BORDER\_STYLE\_SOLID A single line segment GLXC.BORDER\_STYLE\_INSET Looks as if the content is sunken into the canvas GLXC.BORDER\_STYLE\_OUTSET Looks as if the content is coming out of the canvas GLXC.BORDER\_STYLE\_HIDDEN Same as `glxc.BORDER_STYLE_NONE` GLXC.BORDER\_STYLE\_DOTTED A series of round dots GLXC.BORDER\_STYLE\_DASHED A series of square-ended dashes GLXC.BORDER\_STYLE\_DOUBLE Two parallel lines with some space between them GLXC.BORDER\_STYLE\_GROOVE Looks as if it were carved in the canvas GLXC.BORDER\_STYLE\_RIDGE Looks as if it were coming out of the canvas

**Returns** a GLXC.BorderStyle type Constant or GLXC.BORDER\_STYLE\_NONE if none was set

**Return type** str

**get\_width\_chars** ()

Gets the value set by `set_width_chars()`

**Returns** number of chars to request space for, or negative if unset

**get\_max\_width\_chars()**

Retrieves the desired maximum width of entry, in characters.

set\_max\_width\_chars().

**Returns** the maximum width of the entry, in characters

**Return type** int

**set\_activates\_default(setting)**

If setting is True, pressing Enter in the entry will activate the default widget for the window containing the entry.

This usually means that the dialog box containing the entry will be closed, since the default widget is usually one of the dialog buttons.

(For experts: if setting is True, the entry calls activate\_default() on the window containing the entry, in the default handler for the “activate” signal.)

**Parameters** **setting** (*bool*) – True to activate window’s default widget on Enter keypress

**Raises** **TypeError** – if setting is not bool type

**set\_has\_frame(setting=True)**

Sets whether the entry has a beveled frame around it.

**Parameters** **setting** (*bool*) – False removes outside bevel from entry

**Raises** **TypeError** – if setting is not bool type

**set\_inner\_border(border='BORDER\_STYLE\_NONE')**

Sets entry’s inner-border property to border, or clears it if None is passed. The inner-border is the area around the entry’s text, but inside its frame.

If set, this property overrides the inner-border style property. Overriding the style-provided border is useful when you want to do in-place editing of some text in a canvas or list widget, where pixel-exact positioning of the entry is important.

**GLXC.BorderStyle**

Describes how the border of a UI element should be rendered.

**Members:** GLXC.BORDER\_STYLE\_NONE No visible border GLXC.BORDER\_STYLE\_SOLID A single line segment GLXC.BORDER\_STYLE\_INSET Looks as if the content is sunken into the canvas GLXC.BORDER\_STYLE\_OUTSET Looks as if the content is coming out of the canvas GLXC.BORDER\_STYLE\_HIDDEN Same as glxc.BORDER\_STYLE\_NONE GLXC.BORDER\_STYLE\_DOTTED A series of round dots GLXC.BORDER\_STYLE\_DASHED A series of square-ended dashes GLXC.BORDER\_STYLE\_DOUBLE Two parallel lines with some space between them GLXC.BORDER\_STYLE\_GROOVE Looks as if it were carved in the canvas GLXC.BORDER\_STYLE\_RIDGE Looks as if it were coming out of the canvas

**Parameters** **border** (*str*) – a valid GLXC.BorderStyle

**Raises**

- **TypeError** – if border is not str type
- **TypeError** – if border is not a valid GLXC.BorderStyle

**set\_width\_chars(n\_chars=-1)**

Changes the size request of the entry to be about the right size for n\_chars characters. Note that it changes the size request, the size can still be affected by how you pack the widget into containers.

If n\_chars is -1, the size reverts to the default entry size.

**Parameters** `n_chars` (*int*) – width in chars

**Raises** **TypeError** – if `n_chars` is not int type

**set\_max\_width\_chars** (`n_chars=-1`)

Sets the desired maximum width in characters of entry

**Parameters** `n_chars` (*int*) – the new desired maximum width, in characters

**Raises** **TypeError** – if `n_chars` is not int type

**get\_invisible\_char** ()

Retrieves the character displayed in place of the real characters for entries with visibility set to false.

**See also:**

`set_invisible_char()`.

**Returns** the current invisible char, or 0, if the entry does not show invisible text at all.

**set\_alignment** (`xalign=0.0`)

Sets the alignment for the contents of the entry. This controls the horizontal positioning of the contents when the displayed text is shorter than the width of the entry.

**Parameters** `xalign` (*float*) – The horizontal alignment, from 0 (left) to 1 (right). Reversed for RTL layouts

**Raises** **TypeError** – if `xalign` is not float type

**get\_alignment** ()

Gets the value set by `GLXCurses.Entry.set_alignment()`.

**Returns** The horizontal alignment, from 0 (left) to 1 (right).

**Return type** float

**set\_placeholder\_text** (`text=None`)

Sets text to be displayed in entry when it is empty and unfocused. This can be used to give a visual hint of the expected contents of the `GLXCurses.Entry`.

---

**Note:** that since the placeholder text gets removed when the entry received focus, using this feature is a bit problematic if the entry is given the initial focus in a window. Sometimes this can be worked around by delaying the initial focus setting until the first key event arrives.

---

**Parameters** `text` (*str or None*) – a string to be displayed when entry is empty and unfocused, or None.

**Raises** **TypeError** – if `text` is not str or None type

**get\_placeholder\_text** ()

Retrieves the text that will be displayed when entry is empty and unfocused

**Returns** a pointer to the placeholder text as a string. This string points to internally allocated storage in the widget and must not be freed, modified or stored.

**set\_overwrite\_mode** (`overwrite=False`)

Sets whether the text is overwritten when typing in the `GLXCurses.Entry`.

**Parameters** `overwrite` (*bool*) – new value

**Raises** **TypeError** – if `overwrite` is not bool type

**get\_overwrite\_mode()**

Gets the value set by `GLXCurses.Entry.set_overwrite_mode()`.

**Returns** whether the text is overwritten when typing.

**Return type** bool

**get\_layout()**

**Raises** **NotImplementedError** – GLXCurses don't get Pango management

**get\_layout\_offsets()**

**Raises** **NotImplementedError** – GLXCurses don't get Pango management

**layout\_index\_to\_text\_index()**

**Raises** **NotImplementedError** – GLXCurses don't get Pango management

**text\_index\_to\_layout\_index()**

**Raises** **NotImplementedError** – GLXCurses don't get Pango management

**set\_attributes()**

**Raises** **NotImplementedError** – GLXCurses don't get Pango management

**get\_attributes()**

**Raises** **NotImplementedError** – GLXCurses don't get Pango management

**get\_max\_length()**

Retrieves the maximum allowed length of the text in entry . See `GLXCurses.Entry.set_max_length()`.

This is equivalent to getting entry 's `GLXCurses.EntryBuffer` and calling `GLXCurses.EntryBuffer.get_max_length()` on it.

**Returns** the maximum allowed number of characters in `GLXCurses.Entry`, or 0 if there is no maximum.

**Return type** int

**get\_visibility()**

Retrieves whether the text in entry is visible.

---

**Note:** `GLXCurses.EntryBuffer.set_visibility()`

---

**Returns** True if the text is currently visible

**Return type** bool

**set\_completion(completion=None)**

Sets completion to be the auxiliary completion object to use with entry . All further configuration of the completion mechanism is done on completion using the `GtkEntryCompletion` API. Completion is disabled if completion is set to None.

**Parameters** **completion** (`GLXCurses.EntryCompletion.EntryCompletion` or `None`) – The `GLXCurses.EntryCompletion.EntryCompletion` or None.

**Raises** **TypeError** – when completion is not `GLXCurses.EntryCompletion` or None

**get\_completion()**

Returns the auxiliary completion object currently in use by entry .

**Returns** The auxiliary completion object currently in use by entry .

**Return type** GLXCurses.EntryCompletion or *None*

**set\_cursor\_hadjustment** (*adjustment=None*)

Hooks up an adjustment to the cursor position in an entry, so that when the cursor is moved, the adjustment is scrolled to show that position. See `scrolled_window_get_hadjustment()` for a typical way of obtaining the adjustment.

The adjustment has to be in char units and in the same coordinate system as the entry.

**Parameters** **adjustment** (`GLXCurses.Adjustment.Adjustment` or *None*) – an adjustment which should be adjusted when the cursor is moved, or *None*.

**Raises** **TypeError** – when completion is not `GLXCurses.Adjustment.Adjustment` or *None*

**get\_cursor\_hadjustment** ()

Retrieves the horizontal cursor adjustment for the entry. See `GLXCurses.Adjustment.Adjustment.set_cursor_hadjustment()`.

**Returns** the horizontal cursor adjustment, or *NULL* if none has been set.

**Return type** `GLXCurses.Adjustment.Adjustment` or *None*

**set\_progress\_fraction** (*fraction=0.0*)

Causes the entry’s progress indicator to “fill in” the given fraction of the bar. The fraction should be between 0.0 and 1.0, inclusive.

**Parameters** **fraction** (*float*) – fraction of the task that’s been completed

**Raises** **TypeError** – when fraction is not float type

**get\_progress\_fraction** ()

Returns the current fraction of the task that’s been completed. See `GLXCurses.Entry.Entry.set_progress_fraction()`.

**Returns** a fraction from 0.0 to 1.0

**Return type** float

**set\_progress\_pulse\_step** (*fraction=0.1*)

Sets the fraction of total entry width to move the progress bouncing block for each call to `GLXCurses.Entry.Entry.progress_pulse()`.

**Parameters** **fraction** (*float*) – fraction between 0.0 and 1.0

**Raises** **TypeError** – when fraction is not float type

**get\_progress\_pulse\_step** ()

Retrieves the pulse step set with `GLXCurses.Entry.Entry.set_progress_pulse_step()`.

**Returns** a fraction from 0.0 to 1.0

**Return type** float

**progress\_pulse** ()

Indicates that some progress is made, but you don’t know how much. Causes the entry’s progress indicator to enter “activity mode,” where a block bounces back and forth. Each call to `GLXCurses.Entry.Entry.progress_pulse()` causes the block to move by a little bit (the amount of movement per pulse is determined by `GLXCurses.Entry.Entry.set_progress_pulse_step()`).

**Raises** **NotImplementedError** – `GLXCurses` don’t deal with that yet.

**im\_context\_filter\_keypress** ()

Allow the `GLXCurses.Entry` input method to internally handle key press and release events.

If this function returns TRUE, then no further processing should be done for this key event.

See `GLXCurses.Entry.im_context_filter_keypress()`.

Note that you are expected to call this function from your handler when overriding key event handling. This is needed in the case when you need to insert your own key handling between the input method and the default key event handling of the `GLXCurses.Entry`.

See `GLXCurses.Entry.text_view_reset_im_context()` for an example of use.

**Raises `NotImplementedError`** – `GLXCurses` don't deal with that yet.

**`reset_im_context()`**

Reset the input method context of the entry if needed.

This can be necessary in the case where modifying the buffer would confuse on-going input method behavior.

**Raises `NotImplementedError`** – `GLXCurses` don't deal with that yet.

**`get_tabs()`**

Gets the tabstops that were set on the entry using `gtk_entry_set_tabs()`, if any.

**Raises `NotImplementedError`** – `GLXCurses` don't deal with that yet.

**`set_tabs()`**

Sets a `PangoTabArray`; the tabstops in the array are applied to the entry text.

**Raises `NotImplementedError`** – `GLXCurses` don't deal with that yet.

**`set_icon_from_pixbuf()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management

**`set_icon_from_stock()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management

**`set_icon_from_icon_name()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management

**`set_icon_from_gicon()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management

**`get_icon_storage_type()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management

**`get_icon_pixbuf()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management

**`get_icon_stock()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management

**`get_icon_name()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management

**`get_icon_gicon()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management

**`set_icon_activatable()`**

**Raises `NotImplementedError`** – `GLXCurses` don't get icon's management



`get_icon_activatable()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`set_icon_sensitive()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`get_icon_sensitive()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`get_icon_at_pos()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`set_icon_tooltip_text()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`get_icon_tooltip_text()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`set_icon_tooltip_markup()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`get_icon_tooltip_markup()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`set_icon_drag_source()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`get_current_icon_drag_source()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`get_icon_area()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`set_input_purpose()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`get_input_purpose()`

Raises **NotImplementedError** – GLXCurses don't get icon's management

`set_input_hints()`

Raises **NotImplementedError** – GLXCurses don't deal with hints

`get_input_hints()`

Raises **NotImplementedError** – GLXCurses don't deal with hints

`grab_focus_without_selecting()`

Causes entry to have keyboard focus.

It behaves like `gtk_widget_grab_focus()`, except that it doesn't select the contents of the entry. You only want to call this on some special entries which the user usually doesn't want to replace all text in, such as search-as-you-type entries.

`get_states()`

`update_preferred_sizes()`

**GLXCurses.EntryBuffer module****class** GLXCurses.EntryBuffer.**EntryBuffer**Bases: *GLXCurses.Object.Object*EntryBuffer — Text buffer for *GLXCurses.Entry***Description**

The *GLXCurses.EntryBuffer* class contains the actual text displayed in a *GLXCurses.Entry* widget.

A single *GLXCurses.EntryBuffer* object can be shared by multiple *GLXCurses.Entry* widgets which will then share the same text content, but not the cursor position, visibility attributes, etc.

*GLXCurses.EntryBuffer* may be derived from. Such a derived class might allow text to be stored in an alternate location, such as non-pageable memory, useful in the case of important passwords. Or a derived class could integrate with an application's concept of undo/redo.

**max\_length****length**

The length property

Allowed values: &lt;= 65535

Default value: 0

**Returns** The length (in characters) of the text in buffer.**Return type** int**text**

The text property

**Returns** The contents of the buffer.**Return type** char**new** (*initial\_chars=None, n\_initial\_chars=-1*)Create a new *GLXCurses.EntryBuffer* object.

Optionally, specify initial text to set in the buffer.

**Parameters**

- **initial\_chars** – initial buffer text, or None
- **n\_initial\_chars** – number of characters in initial\_chars , or -1

**Returns** the new EntryBuffer**Return type** *GLXCurses.EntryBuffer.EntryBuffer***Raises**

- **TypeError** – if *initial\_chars* is not printable string or None
- **TypeError** – if *n\_initial\_chars* is not int or -1

**get\_text** ()

Retrieves the contents of the buffer.

The memory pointer returned by this call will not change unless this object emits a signal, or is finalized.

**Returns** a pointer to the contents of the widget as a string. This string points to internally allocated storage in the buffer and must not be freed, modified or stored.

**Return type** str

**set\_text** (*chars*="", *n\_chars*=-1)

Sets the text in the buffer.

This is roughly equivalent to calling `EntryBuffer.delete_text()` and `EntryBuffer.insert_text()`.

---

**Note:** *n\_chars* is in characters, not in bytes.

---

#### Parameters

- **chars** (*str*) – the new text
- **n\_chars** (*int*) – the number of characters in text , or -1

#### Raises

- **TypeError** – if *chars* is not *str*
- **TypeError** – if *n\_chars* is not *int* or -1

**get\_bytes** ()

Retrieves the length in bytes of the buffer.

#### See also:

`EntryBuffer.get_length()`.

**Returns** The byte length of the buffer.

**Return type** *int*

**get\_length** ()

Retrieves the length in characters of the buffer.

**Returns** The number of characters in the buffer.

**Return type** *int*

**get\_max\_length** ()

Retrieves the maximum allowed length of the text in buffer .

#### See also:

`EntryBuffer.set_max_length()`.

**Returns** the maximum allowed number of characters in `EntryBuffer`, or 0 if there is no maximum.

**Return type** *int*

**set\_max\_length** (*max\_length*=0)

Sets the maximum allowed length of the contents of the buffer. If the current contents are longer than the given length, then they will be truncated to fit.

**Parameters** **max\_length** (*int*) – The maximum length of the entry buffer, or 0 for no maximum. (other than the maximum length of entries.) The value passed in will be clamped to the range 0-65536.

**Raises** **TypeError** – if *max\_length* is not *int*

**insert\_text** (*position=0, chars="", n\_chars=-1*)

Inserts `n_chars` characters of `chars` into the contents of the buffer, at position `position`.

If `n_chars` is negative, then characters from `chars` will be inserted until a null-terminator is found. If `position` or `n_chars` are out of bounds, or the maximum buffer text length is exceeded, then they are coerced to sane values.

---

**Note:** The position and length are in characters, not in bytes.

---

#### Parameters

- **position** (*int*) – The position at which to insert text.
- **chars** (*str*) – The text to insert into the buffer.
- **n\_chars** (*int*) – The length of the text in characters, or -1

**Returns** The number of characters actually inserted.

**Return type** `int`

#### Raises

- **TypeError** – if `position` is not `int`
- **TypeError** – if `chars` is not printable `str`
- **TypeError** – if `n_chars` is not `int`

**delete\_text** (*position=None, n\_chars=-1*)

Deletes a sequence of characters from the buffer. `n_chars` characters are deleted starting at `position`. If `n_chars` is negative, then all characters until the end of the text are deleted.

If `position` or `n_chars` are out of bounds, then they are coerced to sane values.

---

**Note:** The positions are specified in characters, not bytes..

---

#### Parameters

- **position** (*int*) – Position at which to delete text
- **n\_chars** (*int*) – Number of characters to delete

**Returns** The number of characters deleted.

**Return type** `int`

#### Raises

- **TypeError** – if `position` is not `int`
- **TypeError** – if `n_chars` is not `int`

## GLXCurses.EntryCompletion module

**class** `GLXCurses.EntryCompletion.EntryCompletion`

Bases: `GLXCurses.Object.Object`

`EntryCompletion` — Completion functionality for `GLXCurses.Entry`

**Properties****inline\_completion**

WDetermines whether the common prefix of the possible completions should be inserted automatically in the entry. Note that this requires text-column to be set, even if you are using a custom match function.

**Type** bool

**Flags** Read / Write

**Default value** False

**inline\_selection**

Determines whether the possible completions on the popup will appear in the entry as you navigate through them.

**Type** bool

**Flags** Read / Write

**Default value** False

**minimum\_key\_length**

Minimum length of the search key in order to look up matches.

**Type** bool

**Flags** Read / Write

**Allowed values**  $\geq 0$

**Default value** 1

**model**

The model to find matches in.

**Type** TreeModel

**Flags** Read / Write

**popup\_completion**

Determines whether the possible completions should be shown in a popup window.

**Type** bool

**Flags** Read / Write

**Default value** True

**popup\_single\_match**

Determines whether the completions popup window will shown for a single possible completion. You probably want to set this to False if you are using inline completion.

**Type** bool

**Flags** Read / Write

**Default value** True

**text\_column**

The column of the model containing the strings. Note that the strings must be UTF-8.

**Type** int

**Flags** Read / Write

**Allowed values**  $\geq -1$

**Default value** -1

**new()**

Creates a new EntryCompletion object.

**Returns** A new GLXCurse Entry Completion object

**Return type** GLXCurse.EntryCompletion

## GLXCurses.EventList module

**class** GLXCurses.EventList.**EventList** (*buffer=None, debug=None*)

Bases: object

**debug**

**buffer**

Return the event\_buffer list attribute, it lis can be edited or modify as you need

**Returns** event buffer

**Return type** list()

**add** (*signal, args*)

Emit a signal, it consist to add the signal structure inside a global event list

**Parameters**

- **signal** – a string containing the signal name
- **args** – additional parameters arg1, arg2

**pop()**

## GLXCurses.FileChooser module

**class** GLXCurses.FileChooser.**FileSelect**

Bases: *GLXCurses.Widget.Widget, GLXCurses.libs.FileChooserFunctions.FileChooserUtils*

**item\_it\_can\_be\_display**

Get the number of item it can be display, as set by \_set\_item\_it\_can\_be\_display().

**Returns** The number of item it can be display

**Return type** int

**item\_scroll\_pos**

Get the number of item it can be display, as set by \_set\_item\_it\_can\_be\_display().

**Returns** The Position on the scroll list

**Return type** int

**selected\_item\_pos**

Position of the selected item.

**Returns** The Position on the scroll list

**Return type** int

**selected\_item\_info\_list**

Get the selected file information's list.

**The line\_info information's store position:** item\_name\_text in position [0] item\_path\_sys in position [1] item\_size\_text in position [2] item\_time\_text in position [3]

**Returns** information's about selected item.

**Return type** dict

```

x_pos_history_next_label
x_pos_history_list_label
x_pos_history_prev_label
x_pos_history_actual_path
x_pos_history_actual_path_allowed_size
x_pos_title_mtime
x_pos_title_size
x_pos_title_name
x_pos_line_start
x_pos_line_stop
y_pos_history
y_pos_titles
y_pos_items
name_column_width
mtime_column_width
size_column_width
draw_widget_in_area()
    Be here for be overwrite by every widget
update_preferred_sizes()

```

## GLXCurses.FileChooserMenu module

```

class GLXCurses.FileChooserMenu.FileChooserMenu(parent=None, y=None, x=None, la-
                                                bel=None)

```

Bases: *GLXCurses.Container.Container, GLXCurses.libs.Movable.Movable*

### Parameters

- **parent** (*Filechooser*) –
- **y** –
- **x** –
- **label** –

```

draw_widget_in_area()
    Be here for be overwrite by every widget

```

**label**

**history\_box\_num\_cols**

```
history_dir_list
draw_titles()
update_size()
```

## GLXCurses.Frame module

```
class GLXCurses.Frame.Frame
    Bases: GLXCurses.Bin.Bin
```

### Description

The frame widget is a bin that surrounds its child with a decorative frame and an optional label. If present, the label is drawn in a gap in the top side of the frame.

The position of the label can be controlled with `Frame.set_label_align()`.

### label

Text of the frame's label.

Default value: None

**Returns** the label property value

**Return type** str or *None*

### label\_widget

A widget to display in place of the usual frame label.

**Returns** A widget

**Return type** GLXCurses.Label or *None*

### label\_xalign

The horizontal alignment of the label.

**Returns** The horizontal alignment of the label.

**Return type** float

### label\_yalign

The vertical alignment of the label.

**Returns** The vertical alignment of the label.

**Return type** float

### shadow\_type

Appearance of the frame border.

**Returns** The shadow type use by the frame

**Return type** GLXCurses.GLXC.ShadowType

### new (label=None)

Create a new *Frame*, with optional label text .

If label is None, the label is omitted.

**Parameters** **label** (*str* or *None*) – the text to use as the label of the frame.

**Returns** a new *Frame* widget

**Return type** *Widget*



**set\_label** (*label*)

Sets the text of the label.

If label is `None`, the current label is removed.

**Parameters** **label** (*str* or `None`) – the text to use as the label of the frame.

**set\_label\_widget** (*label\_widget*)

Sets the label widget for the frame. This is the widget that will appear embedded in the top edge of the frame as a title.

**Parameters** **label\_widget** (*Widget*) – the new label widget

**set\_label\_align** (*xalign*, *yalign*)

Sets the alignment of the frame widget's label. The default values for a newly created frame are 0.0 and 0.5.

**Parameters**

- **xalign** (*float*) – The position of the label along the top edge of the widget. A value of 0.0 represents left alignment; 1.0 represents right alignment.
- **yalign** (*float*) – The y alignment of the label. A value of 0.0 aligns under the frame; 1.0 aligns above the frame. If the values are exactly 0.0 or 1.0 the gap in the frame won't be painted because the label will be completely above or below the frame.

**set\_shadow\_type** (*shadow\_type=None*)

Sets the shadow type for frame .

**Parameters** **shadow\_type** – the new `:py:__area_data:ShadowType`

**get\_label** ()

If the frame's label widget is a `Label`, returns the text in the label widget. (The frame will have a `Label` for the label widget if a non-NULL argument was passed when create the `Frame`.)

**Returns** the text in the label, or `:py:__area_data:None` if there was no label widget or the label widget was not a `Label`. This string is owned by GLXCurses and must not be modified or freed.

**Return type** `str` or `None`

**get\_label\_align** ()

Retrieves the X and Y alignment of the frame's label.

**See also:**

`Frame.set_label_align()`

**xalign**: X location of frame label

**yalign**: Y location of frame label

**Returns** `xalign`, `yalign`

**Return type** `float`, `float`

**get\_label\_widget** ()

Retrieves the label widget for the frame.

**See also:**

`Frame.set_label_widget()`

**Returns** the label widget, or `NULL` if there is none.

**Return type** `Widget` or `:py:__area_data:None`

**get\_shadow\_type()**

Retrieves the shadow type of the frame.

**See also:**

*Frame.set\_shadow\_type()*

**Returns** the current shadow type of the frame.

**Return type** ShadowType

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**update\_preferred\_sizes()**

## GLXCurses.HBox module

**class** GLXCurses.HBox.HBox

Bases: *GLXCurses.Box.Box*, *GLXCurses.libs.Dividable.Dividable*

### Description

The *HBox* is a container that organizes child widgets into a single row.

Use the *Box* packing interface to determine the arrangement, spacing, width, and alignment of *HBox* children.

All children are allocated the same height.

**new** (*homogeneous=True*, *spacing=None*)

Creates a new GLXCurses *HBox*

### Parameters

- **homogeneous** (*bool*) – True if all children are to be given equal space allotments.
- **spacing** (*int*) – The number of characters to place by default between children.

**Returns** a new *HBox*.

### Raises

- **TypeError** – if *homogeneous* is not bool type
- **TypeError** – if *spacing* is not int type or None

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**update\_preferred\_sizes()**

## GLXCurses.HSeparator module

**class** GLXCurses.HSeparator.HSeparator

Bases: *GLXCurses.Widget.Widget*, *GLXCurses.libs.Movable.Movable*

The GLXCurses.HSeparator widget is a horizontal separator, used to visibly separate the widgets within a window.

It displays a horizontal line.

```

draw_widget_in_area()
    Be here for be overwrite by every widget

update_preferred_sizes()

```

## GLXCurses.Image module

```

class GLXCurses.Image.Image
    Bases: GLXCurses.Misc.Misc, GLXCurses.libs.File.File, GLXCurses.libs.Colors.Colors

    image_object
        Store the modified image

        Returns

    data
        Get data property

        Returns image data as a list

        Return type list

    hsp_debug
        Get hsp_debug property

        Returns image hsp_debug as a list

        Return type list

    width_max
        Get the width_max property value

        Returns width_max property value

        Return type int or None

    width_original
        Get the width_original property value

        Returns width_original property value

        Return type int or None

    height_max
        Get the height_max property value

        Returns height_max property value

        Return type int or None

    height_original
        Get the height_original property value

        it property is use when the widget discover image size

        Returns height_original property value

        Return type int or None

    is_resized
        Whether the image will be resized directly on the widget.

        Returns True or False

```

**Return type** bool

**load\_image** (*path=None*)

**draw\_widget\_in\_area** ()

Be here for be overwrite by every widget

**to\_data** ()

## GLXCurses.Label module

**class** GLXCurses.Label.Label

Bases: *GLXCurses.Misc.Misc*, *GLXCurses.libs.Movable.Movable*

**angle**

The angle that the baseline of the label makes with the horizontal, in degrees, measured counterclockwise. An angle of 90 reads from from bottom to top, an angle of 270, from top to bottom.

Ignored if the label is selectable.

Allowed values: [0,360]

**Returns** angle that the baseline of the label

**Return type** int

**attributes**

A list of style attributes to apply to the text of the label.

**Returns** A list of style attributes

**Return type** list

**cursor\_position**

The current position of the insertion cursor in chars.

**Returns** The `cursor_position` property value

**Return type** int

**label**

The contents of the label.

If the string contains TXT Markdown, you will have to set the `use_markdown` property to True in order for the label to display the Markdown attributes.

See also `set_markdown()` for a convenience function that sets both this property and the `use_markdown` property at the same time.

If the string contains underlines acting as mnemonics, you will have to set the `use_underline` property to True in order for the label to display them.

**Returns** The content of the label

**Return type** str

**lines**

The number of lines to which an ellipsized, wrapping label should be limited. This property has no effect if the label is not wrapping or ellipsized.

Set this property to -1 if you don't want to limit the number of lines.

**Returns** The number of lines to which an ellipsized

**Return type** int

**max\_width\_chars**

The desired maximum width of the label, in characters.

If this property is set to -1, the width will be calculated automatically.

See the section on text layout for details of how `width_chars` and `max_width_chars` determine the width of ellipsized and wrapped labels.

**Returns** Maximum width of the label, in characters

**Return type** int

**mnemonic\_keyval**

The mnemonic accelerator key for this label.

Default value: 16777215

**mnemonic\_widget**

The GLXCurses.Widget to be activated when the label's mnemonic key is pressed.

**Returns** The GLXCurses.Widget to be activated or None if not set

:rtype GLXCurses.Widget or None

**pattern**

A string with `_` characters in positions correspond to characters in the text to underline.

**Returns** characters in the text use for underline

**Return type** str

**selectable**

Whether the GLXCurses.Label text can be selected with the mouse.

**Returns** True if GLXCurses.Label text can be selected

**Return type** bool

**selection\_bound**

The position of the opposite end of the selection from the cursor in chars.

**Returns** The position in chars

**Return type** int

**single\_line\_mode**

Whether the label is in single line mode. In single line mode, the height of the label does not depend on the actual text, it is always set to ascent + descent of the font.

This can be an advantage in situations where resizing the label because of text changes would be distracting, e.g. in a GLXCurses.StatusBar or GLXCurses.MessageBar.

Default value: False

**Returns** True if label is in single line mode

**Return type** bool

**track\_visited\_links**

Set this property to True to make the label track which links have been visited.

It will then apply the `GLXC.STATE_FLAG_VISITED` when rendering this link, in addition to `GLXC.STATE_FLAG_LINK`.

Default value: True

**Returns** True if label track which links have been visited

**Return type** bool

**use\_markdown**

The text of the label includes TXT Markdown.

Default value: False

**Returns** True if Markdown is used

**Return type** bool

**use\_underline**

If set, an underline in the text indicates the next character should be used for the mnemonic accelerator key.

Default value: False

**Returns** True if underline is display on text when use a mnemonic accelerator key

**Return type** bool

**width\_chars**

The desired width of the label, in characters. If this property is set to -1, the width will be calculated automatically.

See the section on text layout for details of how `width_chars` and `max_width_chars` determine the width of ellipsized and wrapped labels.

**Returns** The desired width of the label, in characters.

**Return type** int

**wrap**

If set, wrap lines if the text becomes too wide.

**Returns** True if wrap is in use

**Return type** bool

**wrap\_mode**

If line wrapping is on (see the `wrap` property) this controls how the line wrapping is done.

The default is `GLXC.WRAP_WORD`, which means wrap on word boundaries.

**Returns** How the line wrapping is done

**Return type** `GLXCurses.GLXC.WrapMode`

**new** (*string=None*)

Creates a new label with the given text inside it.

You can pass `None` to get an empty `GLXCurses.Label`.

**Parameters** **string** (*str or None*) – The text of the `GLXCurses.Label`.

**Returns** The new `GLXCurses.Label` it self

**Return type** `GLXCurses.Label`

**set\_text** (*string=None*)

Sets the text within the `GtkLabel` widget. It overwrites any text that was there before.

This function will clear any previously set mnemonic accelerators, and set the `use_underline` property to False as a side effect.

This function will set the `use_markdown` property to False as a side effect.

See also: `GLXCurses.Label().set_markdown()`

**Parameters** `string` (*str* or `None`) – The text you want to set

**set\_attributes** (*attributes=None*)

Sets a `GLXC.StateFlags`; the attributes in the list are applied to the label text.

The attributes set with this function will be applied and merged with any other attributes previously effected by way of the `use_underline` or `use_markup` properties.

While it is not recommended to mix markdown strings with manually set attributes, if you must; know that the attributes will be applied to the label after the markdown string is parsed.

**Parameters** `attributes` (*list* or `None`) – a `GLXC.StateFlags`

**set\_markdown** (*string=None*)

Parses `string` which is marked down with the text markdown language, setting the label's text and attribute list based on the parse results.

This function will set the `use_markup` property to `True` as a side effect.

If you set the label contents using the `label` property you should also ensure that you set the `use_markup` property accordingly.

See also: `GLXCurses.Label().set_text()`

**Parameters** `string` (*str*) – A markdown string (see text markdown format)

**set\_markdown\_with\_mnemonic** (*string*)

Parses `string` which is marked down with the text markdown language, setting the label's text and attribute list based on the parse results.

If characters in `string` are preceded by an underscore, they are underlined indicating that they represent a keyboard accelerator called a mnemonic.

The mnemonic key can be used to activate another `GLXCurses.Widget`, chosen automatically, or explicitly using `GLXCurses.Label().set_mnemonic_widget()`.

**Parameters** `string` (*str*) – A markdown string (see text markdown format)

**set\_pattern** (*pattern=None*)

The pattern of underlines you want under the existing text within the `GLXCurses.Label` widget.

For example if the current text of the label says "FooBarBaz" passing a pattern of "\_\_\_" will underline "Foo" and "Baz" but not "Bar".

**Parameters** `pattern` (*str* or `None`) – The pattern as described above.

**set\_justify** (*jtype=None*)

Sets the alignment of the lines in the text of the label relative to each other.

`GLXCurses.GLXC.JUSTIFY_LEFT` is the default value when the widget is first created with `GLXCurses.Label().new()`

If you instead want to set the alignment of the label as a whole, use `GLXCurses.Widget().set_halign()` instead.

`GLXCurses.Label().set_justify()` has no effect on labels containing only a single line.

**Parameters** `jtype` (*str* or `None`) – a `GLXCurses.GLXC.Justification`

**set\_xalign** (*xalign=None*)

Sets the `xalign` property for label .

**Parameters** `xalign` (*float* or `None`) – the new `xalign` value, between 0 and 1

**set\_yalign** (*yalign=None*)

Sets the `yalign` property for label .

**Parameters** `yalign` (*float* or *None*) – the new yalign value, between 0 and 1

**set\_width\_chars** (*n\_chars=None*)

Sets the desired width in characters of label to `n_chars`.

**Parameters** `n_chars` (*int* or *None*) – the new desired width, in characters.

**set\_max\_width\_chars** (*n\_chars*)

Sets the desired maximum width in characters of label to `n_chars`.

**Parameters** `n_chars` (*int* or *None*) – the new desired maximum width, in characters.

**set\_line\_wrap** (*wrap=None*)

Toggles line wrapping within the `GtkLabel` widget. `True` makes it break lines if text exceeds the widget's size. `False` lets the text get cut off by the edge of the widget if it exceeds the widget size.

Note that setting line wrapping to `TRUE` does not make the label wrap at its parent container's width, because `GLXCurses` widgets conceptually can't make their requisition depend on the parent container's size. For a label that wraps at a specific position, set the label's width using `GLXCurses.Widget().set_size_request()`

**Parameters** `wrap` (*bool* or *None*) – `True` if wrap is enable

**set\_line\_wrap\_mode** (*wrap\_mode=None*)

If line wrapping is on (see `GLXCurses.Label().set_line_wrap()`) this controls how the line wrapping is done. The default is `GLXCurses.GLXC.WRAP_WORD` which means wrap on word boundaries.

**Parameters** `wrap_mode` (*str* or *None*) – the line wrapping mode

**set\_lines** (*lines=None*)

Sets the number of lines to which an ellipsized, wrapping label should be limited. This has no effect if the label is not wrapping or ellipsized.

Set this to -1 if you don't want to limit the number of lines.

**Parameters** `lines` (*int* or *None*) – the desired number of lines, or -1

**get\_mnemonic\_keyval** ()

If the label has been set so that it has an mnemonic key this function returns the keyval used for the mnemonic accelerator.

If there is no mnemonic set up it returns `None`.

**Returns** `ord()` keyval usable for accelerators, or `None`

**Return type** `int` or *None*

**get\_selectable** ()

Gets the value set by `GLXCurses.Label().set_selectable()`.

**Returns** `True` if the user can copy text from the label

**get\_text** ()

Fetches the text from a label widget, as displayed on the screen.

This does not include any embedded underlines indicating mnemonics or markdown.

(See `GLXCurses.Label().get_label()`)

**Returns** the text in the label widget. This is the internal string used by the label, and must not be modified.

**Return type** `str` or *None*



**new\_with\_mnemonic** (*string=None*)

Creates a new GtkLabel, containing the text in *str*.

If characters in *str* are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use `'__'` (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `gtk_label_set_mnemonic_widget()`.

If `GLXCurses.Label().set_mnemonic_widget()` is not called, then the first activatable ancestor of the `GLXCurses.Label` will be chosen as the mnemonic widget. For instance, if the label is inside a button or menu item, the button or menu item will automatically become the mnemonic widget and be activated by the mnemonic.

**Parameters** *string* (*str* or *None*) – The text of the label, with an underscore in front of the mnemonic character.

**select\_region** (*start\_offset=None, end\_offset=None*)

Selects a range of characters in the label, if the label is selectable. See `GLXCurses.Label().set_selectable()`.

If the label is not selectable, this function has no effect. If *start\_offset* or *end\_offset* are -1, then the end of the label will be substituted.

**Parameters**

- **start\_offset** (*int*) – start offset (in characters not bytes)
- **end\_offset** (*int*) – end offset (in characters not bytes)

**Raises** `TypeError` – when

**set\_use\_underline** (*setting*)

**get\_use\_underline** ()

**set\_mnemonic\_widget** (*widget*)

**get\_mnemonic\_widget** ()

**set\_selectable** (*setting=None*)

Selectable labels allow the user to select text from the label, for copy-and-paste.

**Parameters** *setting* (*bool* or *None*) – True to allow selecting text in the label

**set\_text\_with\_mnemonic** (*string*)

Sets the label's text from the string *str*. If characters in *str* are preceded by an underscore, they are underlined indicating that they represent a keyboard accelerator called a mnemonic.

The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `GLXCurses.Label().set_mnemonic_widget()`.

**Parameters** *string* (*str*) – a string

**draw\_widget\_in\_area** ()

Be here for be overwrite by every widget

**update\_preferred\_sizes** ()

**get\_justify** ()

Returns the justification of the label.

**See also:**

`Label.set_justify()` for set the justification.

**Returns** the justification

**Return type** GLXCurses.GLXC.Justification

**get\_line\_wrap()**

The `get_line_wrap()` method returns the value of the “wrap” property.

If “wrap” is True the lines in the label are automatically wrapped. See `set_line_wrap()`.

**Returns** True if wrap is enable

**Return type** bool

**get\_width\_chars()**

The `get_width_chars()` method returns the value of the `width-chars` property that specifies the desired width of the label in characters.

**Returns** width of the label in characters

**Return type** int

**set\_single\_line\_mode()** (*single\_line\_mode*)

**get\_single\_line\_mode()**

**get\_max\_width\_chars()**

**get\_line\_wrap\_mode()**

## GLXCurses.MainLoop module

**class** GLXCurses.MainLoop.Singleton (*name, bases, dictionary*)

Bases: type

**class** GLXCurses.MainLoop.MainLoop (*application=None, debug=None*)

Bases: object

### Description

The MainLoop is something close to a infinity loop with a `start()` and `stop()` method

**Methods:** `start()` – start the mainloop `stop()` – stop the mainloop `emit()` – emit a signal

**Warning:** you have to start the mainloop from you application via `MainLoop().start()`

Creates a new MainLoop structure.

**debug**

**instance** = None

**application**

**event\_list**

**running**

**glxcurses\_support**

**start()**

Runs a MainLoop until `quit()` is called on the loop. If this is called for the thread of the loop's , it will process events from the loop, otherwise it will simply wait.

**stop()**

Stops a MainLoop from running. Any calls to run() for the loop will return.

Note that sources that have already been dispatched when quit() is called will still be executed.

**handle\_event** (*event=None*)

## GLXCurses.Menu module

**class** GLXCurses.Menu.Menu

Bases: *GLXCurses.Window.Window, GLXCurses.libs.Movable.Movable, GLXCurses.Box.Box, GLXCurses.MenuShell.MenuShell*

**color**

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**static remove\_accel\_group()**

Not implemented

**Raises NotImplementedError** – because AccelGroup is not implemented

**static add\_accel\_group()**

Not implemented

**Raises NotImplementedError** – because AccelGroup is not implemented

## GLXCurses.MenuBar module

**class** GLXCurses.MenuBar.MenuBar

Bases: *GLXCurses.Box.Box, GLXCurses.libs.Dividable.Dividable, GLXCurses.MenuShell.MenuShell*

**color**

**info\_label**

**selected\_menu**

**selected\_menu\_item**

**draw\_widget\_in\_area()**

White the menubar to the stdscr, the location is imposed to top left corner

**update\_preferred\_sizes()**

## GLXCurses.MenuItem module

**class** GLXCurses.MenuItem.MenuItem

Bases: *GLXCurses.Widget.Widget*

**accel\_path**

Sets the accelerator path of the menu item, through which runtime changes of the menu item's accelerator caused by the user can be identified and saved to persistant storage.

Default value: NULL

**Returns** The accelerator path of the menu item

**Return type** str

**label**

The text for the child label.

Default value: ""

**Returns** child label

**Return type** str

**Raises** **TypeError** – When `label` property value is not str type or None

**right\_justified**

Sets whether the menu item appears justified at the right side of a menu bar.

Default value: False

**Returns** True if the widget appears justified at the right side of a menu bar

**Return type** bool

**text\_short\_cut**

**spacing**

**resized\_text**

**resized\_text\_short\_cut**

**is\_accel**

**accelerator\_size**

**color**

**draw()**

## GLXCurses.MenuShell module

**class** GLXCurses.MenuShell.MenuShell

Bases: *GLXCurses.Container.Container*

**take\_focus**

**append** (*child=None*)

Adds a new GLXCurses.MenuItem to the end of the menu shell's item list.

**Returns** A new GLXCurses.MenuItem to add

## GLXCurses.MessageBar module

**class** GLXCurses.MessageBar.MessageBar

Bases: *GLXCurses.Widget.Widget*

A MessageBar is usually placed along the bottom of an Application. It may provide a regular commentary of the application's status (as is usually the case in a web browser, for example), or may be used to simply output a message when the status changes, (when an upload is complete in an FTP client, for example).

Status bars in GLXCurses maintain a stack of messages. The message at the top of the each bar's stack is the one that will currently be displayed.

Any messages added to a StatusBar's stack must specify a context id that is used to uniquely identify the source of a message. This context id can be generated by `GLXCurses.StatusBar.get_context_id()`, given a message and the StatusBar that it will be added to. Note that messages are stored in a stack, and when choosing which message to display, the stack structure is adhered to, regardless of the context identifier of a message.

One could say that a StatusBar maintains one stack of messages for display purposes, but allows multiple message producers to maintain sub-stacks of the messages they produced (via context ids).

Status bars are created using `GLXCurses.MessageBar.new()`.

Messages are added to the bar's stack with `GLXCurses.MessageBar.push()`.

The message at the top of the stack can be removed using `GLXCurses.MessageBar.pop()`.

A message can be removed from anywhere in the stack if its message id was recorded at the time it was added. This is done using `GLXCurses.MessageBar.remove()`.

**new()**

Creates a new `GLXCurses.MessageBar` ready for messages.

**Returns** the new MessageBar

**Return type** `GLXCurses.MessageBar`

**get\_context\_id**(*context\_description*='Default')

Returns a new context identifier, given a description of the actual context.

**Parameters** **context\_description** (*str*) – textual description of what context the new message is being used in

**Returns** an context\_id generate by `Utils.new_id()`

**Return type** `str`

**Raises** **TypeError** – When context\_description is not a str

**push**(*context\_id*, *text*)

Push a new message onto the MessageBar's stack.

**Parameters**

- **context\_id** (*str*) – a context identifier, as returned by `MessageBar.get_context_id()`
- **text** (*str*) – the message to add to the MessageBar

**Returns** a message identifier that can be used with `MessageBar.remove()`.

**Return type** `str`

**pop**(*context\_id*)

Removes the first message in the MessageBar's stack with the given context id.

Note that this may not change the displayed message, if the message at the top of the stack has a different context id.

**Parameters** **context\_id** (*str*) – a context identifier, as returned by `MessageBar.get_context_id()`

**remove**(*context\_id*, *message\_id*)

Forces the removal of a message from a MessageBar's stack. The exact **context\_id** and **message\_id** must be specified.

**Parameters**

- **context\_id** (*str*) – a context identifier, as returned by `MessageBar.get_context_id()`
- **message\_id** (*str*) – a message identifier, as returned by `MessageBar.push()`

**remove\_all** (*context\_id*)

Forces the removal of all messages from a MessageBar's stack with the exact *context\_id* .

**Parameters** *context\_id* (*str*) – a context identifier, as returned by MessageBar.get\_context\_id()

**draw** ()

Place the status bar from the end of the stdscr by look if it have a tool bar before

**update\_preferred\_sizes** ()

## GLXCurses.Misc module

**class** GLXCurses.Misc.Misc

Bases: *GLXCurses.Widget.Widget*

The Misc widget is an abstract widget which is not useful itself, but is used to derive subclasses which have alignment and padding attributes.

The horizontal and vertical padding attributes allows extra space to be added around the widget.

The horizontal and vertical alignment attributes enable the widget to be positioned within its allocated area. Note that if the widget is added to a container in such a way that it expands automatically to fill its allocated area, the alignment settings will not alter the widget's position.

Note that the desired effect can in most cases be achieved by using the “halign”, “valign” and “margin” properties on the child widget

**Warning:** To reflect this fact, all Misc API will be deprecated soon.

**xalign** – The horizontal alignment, from 0.0 to 1.0

**yalign** – The vertical alignment, from 0.0 to 1.0

**xpad** – The amount of space to add on the left and right of the widget, in characters

**ypad** – The amount of space to add above and below the widget, in characters

**xalign**

The horizontal alignment. A value of 0.0 means left alignment (or right on RTL locales); a value of 1.0 means right alignment (or left on RTL locales).

Allowed values: [0,1] Default value: 0.5

**Returns** The horizontal alignment value.

**Return type** float

**yalign**

The horizontal alignment. A value of 0.0 means left alignment (or right on RTL locales); a value of 1.0 means right alignment (or left on RTL locales).

Allowed values: [0,1] Default value: 0.5

**Returns** The horizontal alignment

**Return type** float

**xpad**

The amount of space to add on the left and right of the widget, in chars.

**Returns** The amount of space in chars

**Return type** int

**ypad**

**Returns** The amount of space in chars

**Raise** int

## GLXCurses.Object module

**class** GLXCurses.Object.Object

Bases: glxeveloop.bus.Bus

### Description

Object is the fundamental type providing the common attributes and methods for all object types in GLXCurses.

The Object class provides methods for object construction and destruction, property access methods, and signal support.

Signals are described in detail here.

**id**

Return the `id` property value

**Returns** a unique id

**Return type** str

**children**

**debug**

**debug\_level**

Get the debugging information's level to display on the stdscr.

Range: 0 to 3

**Returns** The `debug_level` property value

**Return type** int

**flags**

Return the `flags` attribute, it consist to a dictionary it store keys with have special name.

**Returns** a Dictionary with Galaxie Curses Object Flags format

**Return type** dict

**default\_flags**

**destroy()**

Destroy the object

**eveloop\_dispatch** (*detailed\_signal*, *args*)

Inform every children or child about a event and execute a eventual callback

### Parameters

- **detailed\_signal** (*str*) – a string containing the signal name
- **args** (*list*) – additional parameters `arg1`, `arg2`

### GLXCurses.ProgressBar module

```
class GLXCurses.ProgressBar.ProgressBar
    Bases: GLXCurses.Widget.Widget, GLXCurses.libs.Movable.Movable
    color (pos=0)
    text
    value
    show_text
    draw_widget_in_area()
        Be here for be overwrite by every widget
    update_preferred_sizes()
```

### GLXCurses.RadioButton module

```
class GLXCurses.RadioButton.RadioButton
    Bases: GLXCurses.Widget.Widget, GLXCurses.libs.Movable.Movable
    active
    text
    interface
    color
    draw_widget_in_area()
        Be here for be overwrite by every widget
    update_preferred_sizes()
```

### GLXCurses.Range module

```
class GLXCurses.Range.Range
    Bases: GLXCurses.Widget.Widget
    Range — Base class for widgets which visualize an adjustment
    Properties
    adjustment
        The GLXCurses.Adjustment.Adjustment that contains the current value of this range object.
        Type GLXCurses.Adjustment.Adjustment
        Flags Read / Write / Construct
    fill_level
        The fill level (e.g. prebuffering of a network stream). See GLX-
        Curses.Adjustment.Adjustment.set_fill_level().
        Type float
        Flags Read / Write
        Default value 1.79769e+308
```



**inverted**

Invert direction slider moves to increase range value.

**Type** bool

**Flags** Read / Write

**Default value** False

**model**

The model to find matches in.

**Type** TreeModel

**Flags** Read / Write

**lower\_stepper\_sensitivity**

The sensitivity policy for the stepper that points to the adjustment's lower side.

**Type** bool

**Flags** Read / Write

**Default value** GLXCurses.GLXC.SENSITIVITY\_AUTO

**restrict\_to\_fill\_level**

The restrict-to-fill-level property controls whether slider movement is restricted to an upper boundary set by the fill level. See GLXCurses.Adjustment.Adjustment.set\_restrict\_to\_fill\_level().

**Type** bool

**Flags** Read / Write

**Default value** True

**round-digits**

The number of digits to round the value to when it changes, or -1. See "change-value".

**Type** int

**Flags** Read / Write

**Allowed values**  $\geq -1$

**Default value** -1

**show\_fill\_level**

The show-fill-level property controls whether fill level indicator graphics are displayed on the trough. See GLXCurses.Adjustment.Adjustment.set\_show\_fill\_level().

**Type** bool

**Flags** Read / Write

**Default value** False

**upper\_stepper\_sensitivity**

The sensitivity policy for the stepper that points to the adjustment's upper side.

**Type** GLXCurses.GLXC.SensitivityType

**Flags** Read / Write

**Default value** GLXC.SENSITIVITY\_AUTO

**get\_fill\_level ()**

Gets the current position of the fill level indicator.

**Returns** The current fill level

**Return type** int

**get\_restrict\_to\_fill\_level()**

Gets whether the range is restricted to the fill level.

**Returns** True if range is restricted to the fill level.

**Return type** bool

**get\_show\_fill\_level()**

Gets whether the range displays the fill level graphically.

**Returns** True if range shows the fill level.

**Return type** bool

**set\_fill\_level(fill\_level=1.79769e+308)**

Set the new position of the fill level indicator.

The “fill level” is probably best described by its most prominent use case, which is an indicator for the amount of pre-buffering in a streaming media player. In that use case, the value of the range would indicate the current play position, and the fill level would be the position up to which the file/stream has been downloaded.

This amount of prebuffering can be displayed on the range’s trough and is themeable separately from the trough. To enable fill level display, use `GLXCurses.Range.Range.set_show_fill_level()`. The range defaults to not showing the fill level.

Additionally, it’s possible to restrict the range’s slider position to values which are smaller than the fill level. This is controller by `GLXCurses.Range.Range.set_restrict_to_fill_level()` and is by default enabled.

**Parameters** **fill\_level** (*float*) – the new position of the fill level indicator

**Raises** **TypeError** – if `fill_level` is not a float type

**set\_restrict\_to\_fill\_level(restrict\_to\_fill\_level=True)**

Sets whether the slider is restricted to the fill level. See `GLXCurses.Range.Range.set_fill_level()` for a general description of the fill level concept.

**Parameters** **restrict\_to\_fill\_level** (*bool*) – Whether the fill level restricts slider movement.

**Raises** **TypeError** – if `restrict_to_fill_level` is not a bool type

**set\_show\_fill\_level(show\_fill\_level)**

Sets whether a graphical fill level is show on the trough. See `GLXCurses.Range.Range.set_fill_level()` for a general description of the fill level concept.

**Parameters** **show\_fill\_level** (*bool*) – Whether a fill level indicator graphics is shown.

**Raises** **TypeError** – if `show_fill_level` is not a bool type

**get\_adjustment()**

Get the `GLXCurses.Adjustment.Adjustment` which is the “model” object for `GLXCurses.Range.Range`. See `GLXCurses.Range.Range.set_adjustment()` for details.

That because `GLXCurses.Range.Range` use internally a `GLXCurses.Adjustment.Adjustment`, the Attribute `adjustment` should never been touch or unreferenced.

**Returns** A `GLXCurses.Adjustment.Adjustment`

**Return type** *GLXCurses.Adjustment.Adjustment*

**set\_adjustment** (*adjustment=None*)

Sets the adjustment to be used as the “model” object for this range widget.

The adjustment indicates the current range value, the minimum and maximum range values, the step/page increments used for keybindings and scrolling, and the page size. The page size is normally 0 for GtkScale and nonzero for Scrollbar, and indicates the size of the visible area of the widget being scrolled. The page size affects the size of the scrollbar slider.

**Parameters** **adjustment** (`GLXCurses.Adjustment.Adjustment` or `None`) – GLXCurses.Adjustment.Adjustment or None for create a new one

**Raises** **TypeError** – if *adjustment* is not a GLXCurses.Adjustment.Adjustment or None

**get\_inverted** ()

Gets the value set by GLXCurses.Range.Range.set\_inverted().

**Returns** True if the range is inverted

**Return type** bool

**set\_inverted** (*setting=False*)

Ranges normally move from lower to higher values as the slider moves from top to bottom or left to right. Inverted ranges have higher values at the top or on the right rather than on the bottom or left.

**Parameters** **setting** (*bool*) – True to invert the range

**Raises** **TypeError** – if *setting* is not a bool type

**get\_value** ()

Gets the current value of the range.

**Returns** current value of the range.

**Return type** float

**set\_value** (*value=<class 'float'>*)

Sets the current value of the range; if the value is outside the minimum or maximum range values, it will be clamped to fit inside them. The range emits the “value-changed” signal if the value changes.

**Parameters** **value** (*float*) – new value of the range

**Raises** **TypeError** – if *value* is not a float type

**set\_increments** (*step=<class 'float'>, page=<class 'float'>*)

Sets the step increment and page increment for the range. The step increment is used when the user clicks the GLXCurses.Scrollbar.Scrollbar arrows or moves GLXCurses.Scale.Scale via arrow keys. The page size is used for example when moving via Page Up or Page Down keys.

Care: the GTK documentation is wrong compare to the the GTK Code source: <https://github.com/GNOME/gtk/blob/master/gtk/gtkrange.c#L1001>

That is step\_increment and page\_increment it be upgrade via a Adjustment.configure() and not step size and page size.

**Parameters**

- **step** (*float*) – the new step increment
- **page** (*float*) – the new page increment

**set\_range** (*min=None, max=None*)

Sets the allowable values in the GLXCurses.Range.Range, and clamps the range value to be between min and max . (If the range has a non-zero page size, it is clamped between min and max - page-size.)

**Parameters**

- **min** (*float*) – minimum range value
- **max** (*float*) – maximum range value

**get\_round\_digits** ()

Gets the number of digits to round the value to when it changes. See “change-value”.

**Returns** the number of digits to round to

**Return type** int

**set\_round\_digits** (*round\_digits=-1*)

Sets the number of digits to round the value to when it changes. See “change-value”.

**Parameters** **round\_digits** (*int*) – the precision in digits, or -1

**Raises** **TypeError** – if *round\_digits* is not a int type

**set\_lower\_stepper\_sensitivity** (*sensitivity='AUTO'*)

Sets the sensitivity policy for the stepper that points to the ‘lower’ end of the GLXCurses.Range.Range’s adjustment.

Allowed Type:

**The arrow is made insensitive if the thumb is at the end** GLXCurses.GLXC.SENSITIVITY\_AUTO = ‘AUTO’

**The arrow is always sensitive** GLXCurses.GLXC.SENSITIVITY\_ON = ‘ON’

**The arrow is always insensitive** GLXCurses.GLXC.SENSITIVITY\_OFF = ‘OFF’

**Parameters** **sensitivity** (*GLXCurses.GLXC.SensitivityType*) – the lower stepper’s sensitivity policy.

**Raises** **TypeError** – if *sensitivity* is not a GLXCurses.GLXC.SensitivityType

**get\_lower\_stepper\_sensitivity** ()

Gets the sensitivity policy for the stepper that points to the ‘lower’ end of the GLXCurses.Range.Range’s adjustment.

**Returns** The lower stepper’s sensitivity policy.

**Return type** GLXCurses.GLXC.SensitivityType

**set\_upper\_stepper\_sensitivity** (*sensitivity='AUTO'*)

Sets the sensitivity policy for the stepper that points to the ‘upper’ end of the GLXCurses.Range.Range’s adjustment.

**Parameters** **sensitivity** (*GLXCurses.GLXC.SensitivityType*) – The upper stepper’s sensitivity policy.

**Raises** **TypeError** – if *sensitivity* is not a GLXCurses.GLXC.SensitivityType

**get\_upper\_stepper\_sensitivity** ()

Gets the sensitivity policy for the stepper that points to the ‘upper’ end of the GLXCurses.Range.Range’s adjustment.

**Returns** The upper stepper’s sensitivity policy.

**Return type** GLXCurses.GLXC.SensitivityType

**get\_flippable** ()

Gets the value set by GLXCurses.Range.Range.set\_flippable().

**Returns** True if the range is flippable

**Return type** bool

**set\_flippable** (*flippable=False*)

If a range is flippable, it will switch its direction if it is horizontal and its direction is GLXCurses.GLXC.TEXT\_DIR\_RTL.

**Parameters** **flippable** (*bool*) – True to make the range flippable

**Raises** **TypeError** – if `flippable` is not a bool type.

**get\_range\_rect** ()

This function returns the area that contains the range's trough and its steppers, in widget->window coordinates.

This function is useful mainly for Range subclasses.

**Returns** list(x, y, width, height)

**Return type** list

**get\_slider\_range** (*slider\_start=None, slider\_end=None*)

This function returns sliders range along the long dimension, in widget->window coordinates.

This function is useful mainly for Range subclasses.

If `slider_start` or `slider_end` are not None it will return the value.

Example:

`slider_start=None, slider_end=None` return list [None; None]

`slider_start=1, slider_end=1` return list [the\_calculated\_slider\_start; the\_calculated\_slider\_end]

**Parameters**

- **slider\_start** – return location for the slider's start, or None
- **slider\_end** – return location for the slider's end, or None

**get\_slider\_size\_fixed** ()

This function is useful mainly for GtkRange subclasses.

See GLXCurses.Range.Range.set\_slider\_size\_fixed().

**Returns** whether the range's slider has a fixed size.

**Return type** bool

**set\_slider\_size\_fixed** (*size\_fixed=<class 'bool'>*)

Sets whether the range's slider has a fixed size, or a size that depends on its adjustment's page size.

This function is useful mainly for GtkRange subclasses.

**Parameters** **size\_fixed** (*bool*) – True to make the slider size constant

**Raises** **TypeError** – if `size_fixed` is not a bool type.

## GLXCurses.Scrollable module

## GLXCurses.StatusBar module

**class** GLXCurses.StatusBar.StatusBar

Bases: *GLXCurses.Widget.Widget*

A StatusBar is usually placed along the bottom of an Application. It may provide a regular commentary of the application's status (as is usually the case in a web browser, for example), or may be used to simply output a message when the status changes, (when an upload is complete in an FTP client, for example).

Status bars in GLXCurses maintain a stack of messages. The message at the top of the each bar's stack is the one that will currently be displayed.

Any messages added to a StatusBar's stack must specify a context id that is used to uniquely identify the source of a message. This context id can be generated by `GLXCurses.StatusBar.get_context_id()`, given a message and the StatusBar that it will be added to. Note that messages are stored in a stack, and when choosing which message to display, the stack structure is adhered to, regardless of the context identifier of a message.

One could say that a StatusBar maintains one stack of messages for display purposes, but allows multiple message producers to maintain sub-stacks of the messages they produced (via context ids).

Status bars are created using `GLXCurses.StatusBar.new()`.

Messages are added to the bar's stack with `GLXCurses.StatusBar.push()`.

The message at the top of the stack can be removed using `GLXCurses.StatusBar.pop()`.

A message can be removed from anywhere in the stack if its message id was recorded at the time it was added. This is done using `GLXCurses.StatusBar.remove()`.

**new()**

Creates a new `GLXCurses.StatusBar` ready for messages.

**Returns** the new StatusBar

**Return type** `GLXCurses.StatusBar`

**get\_context\_id**(*context\_description*='Default')

Returns a new context identifier, given a description of the actual context.

**Parameters** **context\_description** (*str*) – textual description of what context the new message is being used in. Default if none

**Returns** an context\_id generate by `Utils.new_id()`

**Return type** `str`

**Raises** **TypeError** – When context\_description is not a str

**push**(*context\_id*, *text*)

Push a new message onto the StatusBar's stack.

**Parameters**

- **context\_id** (*str*) – a context identifier, as returned by `StatusBar.get_context_id()`
- **text** (*str*) – the message to add to the StatusBar

**Returns** a message identifier that can be used with `StatusBar.remove()`.

**Return type** `str`

**pop**(*context\_id*)

Removes the first message in the StatusBar's stack with the given context id.

Note that this may not change the displayed message, if the message at the top of the stack has a different context id.

**Parameters** **context\_id** (*str*) – a context identifier, as returned by `StatusBar.get_context_id()`

**remove** (*context\_id*, *message\_id*)

Forces the removal of a message from a StatusBar's stack. The exact **context\_id** and **message\_id** must be specified.

**Parameters**

- **context\_id** (*str*) – a context identifier, as returned by `StatusBar.get_context_id()`
- **message\_id** (*str*) – a message identifier, as returned by `StatusBar.push()`

**remove\_all** (*context\_id*)

Forces the removal of all messages from a StatusBar's stack with the exact *context\_id*.

**Parameters** **context\_id** (*str*) – a context identifier, as returned by `StatusBar.get_context_id()`

**draw** ()

Place the status bar from the end of the stdscr by look if it have a toolbar and a statusbar before

**update\_preferred\_sizes** ()

## GLXCurses.Style module

**class** `GLXCurses.Style.Style`

Bases: `GLXCurses.libs.Colors.Colors`

**Description**

Galaxie Curses Style is equivalent to a skin feature, the entire API receive a common Style from Application and each individual Widget can use it own separate one.

Yet it's a bit hard to explain how create you own Style, in summary it consist to a dict() it have keys with a special name call `Attribute`, inside that dictionary we create a second level of dict() dedicated to store color value of each `States`

**default\_attributes\_states**

Return a default style, that will be use by the entire GLXCurses API via the `__attribute_states` object. every Widget's will receive it style by default.

**Returns** A Galaxie Curses Style dictionary

**Return type** dict

**attributes\_states**

Return the `__attribute_states` attribute, it consist to a dictionary it store a second level of dictionary with keys if have special name.

**Returns** attribute states dictionary on Galaxie Curses Style format

**Return type** dict

**attribute\_to\_rgb** (*attribute*='base', *state*='STATE\_NORMAL')

Return a text color, for a attribute and a state passed as argument, it's use by widget for know which color use, when a state change.

By example: When color change if the button is pressed

**Parameters**

- **attribute** – accepted value: `text_fg`
- **state** – accepted value: `STATE_NORMAL`, `STATE_ACTIVE`, `STATE_PRELIGHT`, `STATE_SELECTED`, `STATE_INSENSITIVE`

**Returns** text color

**Return type** str

## GLXCurses.TextBuffer module

**class** GLXCurses.TextBuffer.TextBuffer

Bases: *GLXCurses.Object.Object*

**cursor\_position**

The position of the insert mark (as offset from the beginning of the buffer). It is useful for getting notified when the cursor moves.

**Returns** The cursor position

**Return type** int

**has\_selection**

Whether the buffer has some text currently selected.

**Returns** True when buffer have selection

**Return type** bool

**text**

## GLXCurses.TextPad module

## GLXCurses.TextTag module

**class** GLXCurses.TextTag.TextTag

Bases: *GLXCurses.Object.Object*

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

Tags should be in the TextTagTable for a given TextBuffer before using them with that buffer.

For each property of TextTag, there is a “set” property, e.g. “font-set” corresponds to “font”. These “set” properties reflect whether a property has been set or not.

They are maintained by GLXCurses and you should not set them independently.

**accumulative\_margin**

Whether the margins accumulate or override each other.

When set to `True` the margins of this tag are added to the margins of any other non-accumulative margins present.

When set to `False` the margins override one another (the default).

Default value is `False` and be restore when `accumulative_margin` is set to `None`

**Returns** If `True` the margins of this tag are added to the margins of any other non-accumulative

**Return type** bool

**background**

Background color as a string.

**Returns** color as a string



**Return type** str

**background\_full\_height**

Whether the background color fills the entire line height or only the height of the tagged characters.

When set to `True` the background color fills the entire line height

Default value is `False` and be restore when `background_full_height` is set to `None`

**Returns** If `True` the background color fills the entire line height

**Return type** bool

**background\_full\_height\_set**

Whether this tag affects background height.

When set to `True` this tag affects background height

Default value is `False` and be restore when `background_full_height_set` is set to `None`

**Returns** `True` If this tag affects background height

**Return type** bool

**background\_rgb**

Background color as a RGB.

Default value is `{'r': 0, 'g': 0, 'b': 255}` and be restore when `background_rgb` is set to `None`

**Returns** The RGB color as dict with `r`, `g`, `b` key\_name

**Return type** dict

**background\_set**

Whether this tag affects the background color.

Default value is `False` and be restore when `background_set` is set to `None`

**Returns** If `True`, this tag affects the background color

**Return type** bool

**direction**

Text direction, e.g. right-to-left -> 'RTL' or left-to-right -> 'LTR'.

**Returns** `GLXC.TextDirection` direction type

**Return type** str

**editable**

Whether the text can be modified by the user.

Default value is `True` and be restore when `editable` is set to `None`

**Returns** If `True`, the text can be modified by the user.

**Return type** bool

**editable\_set**

Whether this tag affects text editability.

Default value is `False` and be restore when `editable_set` is set to `None`

**Returns** If `False`, text editability is disable

**Return type** bool

**family**

Name of the font family, e.g. Sans, Helvetica, Times, Monospace.

Default value is `None` and be restore when `family` is set to `None`

**Returns** The font family name

**Return type** str or *None*

**family\_set**

Whether this tag affects the font family.

Default value is `False` and be restore when `editable_set` is set to `None`

**Returns** If `False`, text editability is disable

**Return type** bool

**GLXCurses.TextTagTable module****class** GLXCurses.TextTagTable.TextTagTable

Bases: *GLXCurses.Object.Object*

**new()**

Creates a new GLXCurses.TextTagTable. The table contains no tags by default.

**Returns** a new GLXCurses.TextTagTable

**Return type** GLXCurses.TextTagTable

**add()****remove()****lookup()****foreach()****get\_size()****GLXCurses.TextView module****class** GLXCurses.TextView.TextView

Bases: *GLXCurses.Container.Container*

**accept\_tab**

Whether Tab will result in a tab character being entered.

Default value is `True`, and be restored when `accept_tab` is set to `None`

**Returns** If `True` Tab key will produce Tab char

**Return type** bool

**bottom\_margin**

The bottom margin for text in the text view.

**Returns** The bottom margin padding

**Return type** int

**buffer**

The buffer which is displayed.

**Returns** a buffer

**Return type** `GLXCurses.TextBuffer`

**cursor\_visible**

If the insertion cursor is shown.

Default value is `True`, and be restored when `cursor_visible` is set to `None`

**Returns** If `True` the cursor will be visible

**Return type** `bool`

**editable**

Whether the text can be modified by the user.

Default value is `True`, and be restored when `editable` is set to `None`

**Returns** If `True` the text can be modified

**Return type** `bool`

**indent**

Amount to indent the paragraph, in chars.

**Returns** indentation in chars

**Return type** `int`

**input\_hints**

Additional hints (beyond “`input_purpose`”) that allow input methods to fine-tune their behaviour.

**Returns** The right margin padding

**Return type** `int`

**input\_purpose**

The purpose of this text field.

This property can be used by on-screen keyboards and other input methods to adjust their behaviour.

**Returns** The right margin padding

**Return type** `int`

**justification**

Left, right, or center justification.

**Returns** `str`

**Return type** `GLXCurses.GLXC.Justification`

**left\_margin**

The left margin for text in the text view.

**Returns** The left margin padding

**Return type** `int`

**overwrite**

Whether entered text overwrites existing contents.

Default value is `False`, and be restored when `overwrite` is set to `None`

**Returns** If `True` text overwrites existing contents

**Return type** `bool`

**populate\_all**

**right\_margin**

The right margin for text in the text view.

**Returns** The right margin padding

**Return type** int

**top\_margin**

The top margin for text in the text view.

**Returns** The top margin padding

**Return type** int

**wrap\_mode****GLXCurses.ToolBar module****class** GLXCurses.ToolBar.ToolBar

Bases: *GLXCurses.Widget.Widget*

**draw()**

Draw the ToolBar widget

**labels**

Get the labels list, it contain items with dictionary with key 'id', 'text', 'end\_coord'

**Returns** The labels list

**Return type** list

**init\_button\_positions()**

Calculate positions of buttons; width is never less than 7

Else distribute the extra width in a way that the middle vertical line (between F5 and F6) aligns with the center of the screen.

The extra width is distributed in this order: F10, F5, F9, F4, ..., F6, F1.

**get\_button\_width** (*i=None*)

return width of one button

**Parameters** *i* (*int*) – button number it start to 0

**Returns** width of one button

**Return type** int

**Raises** **TypeError** – When *i* is not a int type

**get\_button\_by\_x\_coord** (*x=None*)

Return the button number by it X coordinate

**Parameters** *x* (*int*) – X coordinate value

**Returns** the button number

**Return type** int

**Raises** **TypeError** – When *x* is not a int type

**set\_label\_text** (*idx=None, text=None*)

Set the text to a button

**Parameters**

- **idx** (*int*) – The button id it start by 0
- **text** (*str*) – The text to set to it button

**update\_preferred\_sizes** ()

## GLXCurses.VBox module

**class** GLXCurses.VBox.VBox

Bases: *GLXCurses.Box.Box*, *GLXCurses.libs.Dividable.Dividable*

**new** (*homogeneous=True*, *spacing=None*)

Creates a new GLXCurses *VBox*

### Parameters

- **homogeneous** (*bool*) – True if all children are to be given equal space allotments.
- **spacing** (*int*) – The number of characters to place by default between children.

**Returns** a new *VBox*.

### Raises

- **TypeError** – if *homogeneous* is not bool type
- **TypeError** – if *spacing* is not int type or None

**draw\_widget\_in\_area** ()

Be here for be overwrite by every widget

**update\_preferred\_sizes** ()

## GLXCurses.VSeparator module

**class** GLXCurses.VSeparator.VSeparator

Bases: *GLXCurses.Widget.Widget*, *GLXCurses.libs.Movable.Movable*

The GLXCurses.VSeparator widget is a vertical separator, used to visibly separate the widgets within a window.

It displays a vertical line.

**draw\_widget\_in\_area** ()

Be here for be overwrite by every widget

**update\_preferred\_sizes** ()

## GLXCurses.VuMeter module

## GLXCurses.Widget module

**class** GLXCurses.Widget.Widget

Bases: *GLXCurses.Object.Object*, *GLXCurses.Aera.Area*, *GLXCurses.libs.Colorable.Colorable*

**Return type** object

**preferred\_height**

**preferred\_width**

**app\_paintable**

Whether the application will paint directly on the widget.

Owner: Widget Flags: Read / Write Default value: FALSE

**Returns** True or False

**Return type** bool

**can\_default**

Whether the widget can be the default widget.

Owner: Widget Flags: Read / Write Default value: False

**Returns** True if widget can be a default widget, False otherwise

**Return type** bool

**events**

Owner: Widget Flags: Read Default value: False

**Returns** list of flags content in GLXC.EventMask

**Return type** list

**can\_focus**

Whether the widget can accept the input focus.

Owner: Widget Flags: Read / Write Default value: False

**Returns** True or False

**Return type** bool

**can\_prelight**

If True if the widget will display prelight color.

By default that if False, by exemple a container is a hidden Widget and have no raison to display prelight.

At the oposit the prelight of a button can be disable with it property

**Returns** True or False

**Return type** bool

**composite\_child**

Whether the widget is part of a composite widget.

**Returns** True or False

**Return type** bool

**expand**

Whether to expand in both directions. Setting this sets both “hexpand” and “vexpand”

**Returns** True or False

**Return type** bool

**focus\_on\_click**

Whether the widget should grab focus when it is clicked with the mouse.

This property is only relevant for widgets that can take focus.

**Returns** True or False

**Return type** bool

### **halign**

How to distribute horizontal space if widget gets extra space, see GLXC.Align

**Allowed value:**

**Stretch to fill all space if possible, center if no meaningful way to stretch** GLXC.ALIGN\_FILL = 'FILL'

**Snap to left or top side, leaving space on right or bottom** GLXC.ALIGN\_START = 'START'

**Snap to right or bottom side, leaving space on left or top** GLXC.ALIGN\_END = 'END'

**Center natural width of widget inside the allocation** GLXC.ALIGN\_CENTER = 'CENTER'

**Align the widget according to the baseline.** GLXC.ALIGN\_BASELINE = 'BASELINE'

**Returns** a GLXC.Align

**Return type** str

### **has\_default**

Whether the widget is the default widget.

**Returns** True or False

**Return type** bool

### **has\_focus**

Whether the widget has the input focus.

**Returns** True or False

**Return type** bool

### **has\_prelight**

Whether the widget is pre light.

**Returns** True or False

**Return type** bool

### **has\_tooltip**

Enables or disables the emission of “query-tooltip” on widget . A value of `True` indicates that widget can have a tooltip, in this case the widget will be queried using “query-tooltip” to determine whether it will provide a tooltip or not.

**Returns** True or False

**Return type** bool

### **height\_request**

Override for height request of the widget, or -1 if natural request should be used.

**Returns** height\_request property value

**Return type** int

### **hexpand**

Whether to expand horizontally.

**Returns** True if the widget have to expand horizontally

**Return type** bool

**hexpand\_set**

Whether to use the “hexpand” property

**Returns** True if the widget use the “hexpand” property

**Return type** bool

**is\_focus**

Whether the widget is the focus widget within the toplevel.

**Returns** True if the widget is the focus widget within the toplevel.

**Return type** bool

**margin**

All four sides’ margin at once. If read, returns max margin on any side.

Allowed values: [0,32767]

**Returns** max margin on any side

**Return type** int

**margin\_bottom**

This property adds margin outside of the widget’s normal size request, the margin will be added in addition to the size from `Widget.set_size_request()` for example.

Allowed values: [0,32767]

**Returns** Margin on bottom side of widget.

**Return type** int

**margin\_end**

Margin on end of widget, horizontally. This property supports left-to-right and right-to-left text directions.

This property adds margin outside of the widget’s normal size request, the margin will be added in addition to the size from `Widget.set_size_request()` for example.

Allowed values: [0,32767]

**Returns** Margin on end of widget, horizontally.

**Return type** int

**margin\_start**

Margin on start of widget, horizontally. This property supports left-to-right and right-to-left text directions.

This property adds margin outside of the widget’s normal size request, the margin will be added in addition to the size from `Widget.set_size_request()` for example.

Allowed values: [0,32767]

**Returns** Margin on start of widget, horizontally.

**Return type** int

**margin\_top**

Margin on top side of widget.

This property adds margin outside of the widget’s normal size request, the margin will be added in addition to the size from `Widget.set_size_request()` for example.

Allowed values: [0,32767]

**Returns** Margin on top side of widget.

**Return type** int



**name**  
The name of the widget.  
**Returns** name of the widget.  
**Return type** str

**no\_show\_all**  
Whether Widget.show\_all() should not affect this widget.  
**Returns** If True, Widget.show\_all() should not affect this widget  
**Return type** bool

**parent**  
The parent GLXCurses.Container of this GLXCurses.Widget. Must be a GLXCurses.Container.  
**Returns** The parent of the GLXCurses.Widget  
**Return type** GLXCurses.Container

**receives\_default**  
If True, the widget will receive the default action when it is focused.  
**Returns** True if the widget receives default  
**Return type** bool

**sensitive**  
Whether the widget responds to input.  
**Returns** True if teh widget responds to input  
**Return type** bool

**style**  
The style of the widget, which contains information about how it will look (colors, etc).  
**Returns** a GLXCurses.Style instance  
**Return type** GLXCurses.Style

**tooltip\_text**  
This is a convenience property which will take care of getting the tooltip shown if the given string is not NULL: `has-tooltip` will automatically be set to TRUE and there will be taken care of “query-tooltip” in the default signal handler.  
**Returns** tooltip\_text property value  
**Return type** str or *None*

**valign**  
How to distribute vertical space if widget gets extra space, see GLXC.Align  
Default value: GLXC.ALIGN\_FILL  
**Returns** The “valign” property  
**Return type** GLXC.Align

**vexpand**  
Whether to expand vertically. See Widget().set\_vexpand().  
**Returns** True if teh widget have to expand vertically  
**Return type** bool

**vexpand\_set**

Whether to use the “vexpand” property

**Returns** True if the widget use the “vexpand” property

**Return type** bool

**visible**

Whether the widget is visible.

Default value: False

**Returns** True if the widget is visible.

**Return type** bool

**width\_request**

Override for width request of the widget, or -1 if natural request should be used.

**Returns** width\_request property value

**Return type** int

**window**

The widget’s window if it is realized, *None* otherwise.

**Returns** return the Window object if realized, *None* otherwise

**Return type** GLXCurses.Window or *None*

**new ()**

Not totally like GTK yet ...

**Actually:** The Widget.New() “can be” and “is” overridden by each GLXCurses Components.

**Original GTK:** This is a convenience function for creating a widget and setting its properties in one go. For example you might write: `Widget().New(GLXC.TYPE_LABEL, “label”, “Hello World”, “xalign”, 0.0, NULL)` to create a left-aligned label.

**Returns** the GLXCurses.Widget

**Return type** GLXCurses.Widget

**destroy ()**

Destroy the object

**in\_destruction**

Returns whether the widget is currently being destroyed.

This information can sometimes be used to avoid doing unnecessary work.

**Returns** True if widget is being destroyed

**Return type** bool

**destroyed** (*widget=None, widget\_pointer=None*)**unparent** (*widget=None*)

This function is only for use in widget implementations. Should be called by implementations of the remove method on Container, to dissociate a child from the container.

**Parameters** **widget** (*GLXCurses.Widget*) – a GLXCurses.Widget

**Raises**

- **TypeError** – if *widget* is not a valid GLXCurses type.

- **TypeError** – if `widget` is not an instance of `GLXCurses.Widget`.

**show()**

Flags a widget to be displayed. Any widget that isn't shown will not appear on the stdscr.

If you want to show all the widgets in a container, it's easier to call `GLXCurses.Widget.show_all()` on the container, instead of individually showing the widgets.

Remember that you have to show the containers containing a widget, in addition to the widget itself, before it will appear onscreen.

When a toplevel container is shown, it is immediately realized and mapped; other shown widgets are realized and mapped when their toplevel container is realized and mapped.

**show\_now()**

Shows a widget.

If the widget is an unmapped toplevel widget (i.e. a `GLXCurses.Window` that has not yet been shown), enter the main loop and wait for the window to actually be mapped.

Be careful; because the main loop is running, anything can happen during this function.

**hide()**

Reverses the effects of `GLXCurses.Widget.show()`, causing the widget to be hidden (invisible to the user).

**show\_all()**

Recursively shows a widget, and any child widgets (if the widget is a container).

**map**

This function is only for use in widget implementations. Causes a widget to be mapped if it isn't already.

**realize**

Creates the GLXCurses (windowing system) resources associated with a widget. For example, widget->window will be created when a widget is realized. Normally realization happens implicitly; if you show a widget and all its parent containers, then the widget will be realized and mapped automatically.

Realizing a widget requires all the widget's parent widgets to be realized; calling `Widget.realize()` realizes the widget's parents in addition to widget itself. If a widget is not yet inside a toplevel window when you realize it, bad things will happen.

This function is primarily used in widget implementations, and isn't very useful otherwise. Many times when you think you might need it, a better approach is to connect to a signal that will be called after the widget is realized automatically, such as "draw". Or simply `g_signal_connect()` to the "realize" signal.

**Returns** the `realize` property value

**Return type** `bool`

**child\_visible**

The `set_child_visible()` method determines if the widget should be mapped along with its parent.

**Returns**

**get\_toplevel()****set\_decorated(decorated)****get\_decorated()****refresh()****override\_color(color)****override\_background\_color(color)**

**attribute\_states**

Return the `__attribute_states` attribute, it consist to a dictionary it store a second level of dictionary with keys if have special name.

**Returns** attribute states dictionary on Galaxie Curses Style format

**Return type** dict

**has\_window**

Determines whether widget has a GdkWindow of its own.

See `GLXCurses.Widget.set_has_window()`.

**Returns** TRUE if widget has a window, FALSE otherwise

**Return type** bool

**draw()****draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**unchild(widget=None)****GLXCurses.Window module**

**class** `GLXCurses.Window.Window` (*window\_type='TOPLEVEL'*)

Bases: `GLXCurses.Bin.Bin`

Creates a new *Window*, which is a toplevel window that can contain other widgets.

Nearly always, the type of the window should be `GLXC.WINDOW_TOPLEVEL`.

If you're implementing something like a popup menu from scratch (which is a bad idea, just use `Menu`), you might use `GLXC.WINDOW_POPUP`. `GLXC.WINDOW_POPUP` is not for dialogs, though in some other toolkits dialogs are called "popups".

If you simply want an undecorated window (no window borders), use `decorated` property, don't use `GLXC.WINDOW_POPUP`.

**Parameters** *window\_type* (*str*) – type of window contain on `GLXC.WindowType` list

**Returns** a new *Window*.

**Return type** *Window*

**Raises** **TypeError** – if *window\_type* is not in valid `GLXC.WindowType` list

**accept\_focus**

Whether the window should receive the input focus.

Default value: `TRUE`

**Returns** the `accept_focus` property value

**Return type** bool

**Raises** **TypeError** – When `accept_focus` is not a bool type

**application**

The *Application* associated with the window.

The application will be kept alive for at least as long as it has any windows associated with it (see `application_hold()` for a way to keep it alive without windows).

Normally, the connection between the application and the window will remain until the window is destroyed, but you can explicitly remove it by setting the `:application` property to `NULL`.

**Returns** The *Application* associated with the window or `None`

**Return type** *GLXCurses.Application.Application* or *None*

**Raises `TypeError`** – When application property value is not a `GLXCurses.Application` instance

#### **attached\_to**

The widget to which this window is attached. See `GLXCurses.Window().set_attached_to()`.

Examples of places where specifying this relation is useful are for instance a `Menu` created by a `ComboBox`, a completion popup window created by `Entry` or a typeahead search entry created by `TreeView`.

**Returns** The `attached_to` property value

**Return type** `GLXCurses.Widget` or *None*

#### **decorated**

Whether the window should be decorated by the window manager.

Default is `True` :return: the `decorated` property value :rtype: `bool`

#### **default\_height**

The default height of the window, used when initially showing the window.

**Returns** the `default_height` property value

**Return type** `int`

#### **default\_width**

The default width of the window, used when initially showing the window.

**Returns** the `default_width` property value

**Return type** `int`

#### **deletable**

Whether the window frame should have a close button.

**Returns** The `deletable` property value

**Return type** `bool`

#### **destroy\_with\_parent**

Get the `destroy_with_parent` property value

**Returns** `True` if the window will be destroyed when the parent is destroyed.

**Return type** `bool`

#### **focus\_on\_map**

Whether the window should receive the input focus when mapped.

**Returns** the `focus_on_map` property value

**Return type** `bool`

#### **focus\_visible**

Whether ‘focus rectangles’ are currently visible in this window.

**Returns** The `focus_visible` property value

**Return type** `bool`

**gravity**

Window gravity defines the meaning of coordinates passed to `Window.move()`.

See `Window.move()` for more details.

The default window gravity is `GLXC.GRAVITY_NORTH_WEST` which will typically “do what you mean.”

**Returns** window gravity

**Return type** str

**has\_resize\_grip**

Whether the window has a corner resize grip.

Note that the resize grip is only shown if the window is actually resizable and not maximized.

Use “resize-grip-visible” to find out if the resize grip is currently shown.

**Returns** The `has_resize_grip` property value

**Return type** bool

**has\_toplevel\_focus**

Whether the input focus is within this Window.

**Returns** The `has_toplevel_focus` property value

**Return type** bool

**hide\_titlebar\_when\_maximized**

Whether the titlebar should be hidden during maximization.

**Returns** The `hide_titlebar_when_maximized` property value

**Return type** bool

**icon**

Icon for this window.

**Returns** The `icon` property value

**Return type** curses Extended Characters

**icon\_name**

The `icon_name` property specifies the name of the themed icon to use as the window icon.

See `IconTheme` for more details.

**Returns** The `icon_name` property value

**Return type** str or *None*

**is\_active**

Whether the toplevel is the current active window.

**Returns** The `is_active` property value

**Return type** bool

**is\_maximized**

Whether the window is maximized.

**Returns** The `is_maximized` property value

**Return type** bool

**mnemonics\_visible**

Whether mnemonics are currently visible in this window.

This property is maintained by GLXCurses based on user input, and should not be set by applications.

**Returns** The `mnemonics_visible` property value

**Return type** `bool`

**modal**

If `True`, the window is modal (other windows are not usable while this one is up).

**Returns** The `modal` property value

**Return type** `bool`

**resizable**

Gets the value set to `resizable` property.

**Returns** `True` if the user can resize the window

**Return type** `bool`

**role**

Unique identifier for the window to be used when restoring a session.

**Returns** A unique identifier

**Return type** `str` or *None*

**screen**

The screen where this window will be displayed.

**Returns** The screen where this window will be displayed

**Return type** *GLXCurses.Screen* or *None*

**skip\_pager\_hint**

`True` if the window should not be in the pager.

**Returns** The `skip_pager_hint` property value

**Return type** `bool`

**skip\_taskbar\_hint**

`True` if the window should not be in the task bar.

**Returns** The `skip_taskbar_hint` property value

**Return type** `bool`

**startup\_id**

The `startup_id` was originally write-only property for setting window's startup notification identifier.

See `Window.set_startup_id()` for more details.

**Returns** A identifier or *None*

**Return type** `str` or *None*

**title**

The title of the window.

Default value: *None*

**Returns** the `title` property value

**Return type** `str` or *None*

**transient\_for**

Fetches the transient parent for this GLXCurses.Window.

See transient\_for.setter for more details about transient windows.

**Returns** the transient parent for this GLXCurses.Window, or None if no transient parent has been set.

**Return type** GLXCurses.Window or *None*

**type**

Return the type property

**Returns** GLXC.WindowType

**Return type** str

**type\_hint**

Hint to help the desktop environment understand what kind of window this is and how to treat it.

These are hints for the window manager that indicate what type of function the window has.

The window manager can use this when determining decoration and behaviour of the window.

The hint must be set before mapping the window.

**Returns** hint for the window manager

**Return type** str

**urgency\_hint**

True if the window should be brought to the user's attention.

**Returns** the urgency\_hint property value

**Return type** bool

**position**

The initial position of the window.

**Returns** position constraint

**Return type** str

**decoration\_button\_layout**

Decorated button layout property

**Returns** a layout

**Return type** str

**decoration\_resize\_handle**

The decoration\_resize\_handle property

**Returns** Decoration resize handle size.

**Return type** int

**color**

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**static add\_accel\_group()**

Not implemented

**Raises** **NotImplementedError** – because AccelGroup is not implemented



**static remove\_accel\_group()**

Not implemented

**Raises** `NotImplementedError` – because `AccelerGroup` is not implemented

**activate\_focus()**

Activates the current focused widget within the window.

**Returns** True if a widget got activated.

**Return type** bool

**activate\_default()**

Activates the default widget for the window, unless the current focused widget has been configured to receive the default action (see `gtk_widget_set_receives_default()`), in which case the focused widget is activated.

**Returns**

**get\_focus()**

The `get_focus()` method returns the current focused widget within the window.

The focus widget is the widget that would have the focus if the toplevel window is focused.

**Returns** The current focused `GLXCurses.Widget`

**Return type** `GLXCurses.Widget` or *None*

**set\_default(default\_widget=None)**

The default widget is the widget that's activated when the user presses Enter in a dialog (for example). This function sets or unsets the default widget for a *Window*. When setting (rather than unsetting) the default widget it's generally easier to call `Widget.grab_default()` on the widget. Before making a widget the default widget, you must call `Widget.set_can_default()` on the widget you'd like to make the default.

**Parameters** `default_widget` (`GLXCurses.Window`) – a `GLXCurses.Window` or `None` of unset

**Raises** `TypeError` – if `default_widget` is not a `GLXCurses.Widget` instance .

**get\_default\_widget()**

Returns the default widget for window . `GLXCurses.Window().set_default()` for more details.

**Returns** the default `GLXCurses.Widget`, or `None` if there is none.

**Return type** `GLXCurses.Widget` or *None*

**get\_window\_type()**

Gets the type of the window.

Constants.`GLXC.WindowType` are `GLXC.WINDOW_TOPLEVEL` and `GLXC.WINDOW_POPUP`

**Returns** the type of the window

**Return type** `GLXC.WINDOW_TOPLEVEL` or `GLXC.WINDOW_POPUP`

**update\_preferred\_sizes()**

## Module contents

**class** `GLXCurses.Clipboard`

Bases: `object`

**text**

**get ()**

Returns the clipboard object for the given selection.

**Returns** The appropriate clipboard object.

**Return type** *GLXCurses.Clipboard*

**set\_text** (*clipboard=None, text=None, length=-1*)

Sets the contents of the GLXCurses.Clipboard to the given UTF-8 string. GLXCurses will make a copy of the text and take responsibility for responding for requests for the text, and for converting the text into the requested format.

**Parameters**

- **clipboard** (*GLXCurses.Clipboard or None*) – a GLXCurses.Clipboard object or None for self
- **text** (*str or None*) – a UTF-8 string.
- **length** (*int*) – length of text, in bytes, or -1, in which case the length will be determined with len().

**wait\_for\_text** (*clipboard=None*)

Requests the contents of the GLXCurses.Clipboard as text and converts the result to UTF-8 if necessary. This function waits for the \_\_area\_data to be received using the main loop, so events, timeouts, etc, may be dispatched during the wait.

**Parameters** **clipboard** (*GLXCurses.Clipboard*) – a GLXCurses.Clipboard

**Returns** a newly-allocated UTF-8 string or NULL if retrieving the selection \_\_area\_data failed.

This could happen for various reasons, in particular if the clipboard was empty or if the contents of the clipboard could not be converted into text form.).

**Return type** *str*

**set\_can\_store** (*clipboard=None, targets=None, n\_targets=None*)

Hints that the clipboard \_\_area\_data should be stored somewhere when the application exits or when store() is called.

This value is reset when the clipboard owner changes.

**Parameters**

- **clipboard** (*GLXCurses.Clipboard or None*) – a GLXCurses.Clipboard object or None for self
- **targets** (*TYPE Constant or None*) – array containing information about which forms should be stored or None to indicate that all forms should be stored.
- **n\_targets** (*int or None*) – number of elements in targets

**store** (*clipboard=None*)

Stores the current clipboard \_\_area\_data somewhere so that it will stay around after the application has quit.

**Parameters** **clipboard** (*GLXCurses.Clipboard or None*) – a GLXCurses.Clipboard object or None for self

**class** GLXCurses.Screen

Bases: object

**stdscr**

**cbreak**

Normally, the tty driver buffers typed characters until a newline or carriage return is typed. If

`cbreak=True` The `cbreak` property disables line buffering and erase/kill character-processing (interrupt and flow control characters are unaffected), making characters typed by the user immediately available to the program.

The (`cbreak` is `False`) returns the terminal to normal (cooked) mode.

Initially the terminal may or may not be in `cbreak` mode, as the mode is inherited; therefore, a program should call `cbreak=True` or `cbreak=False` explicitly.

Most interactive programs using curses set the `cbreak=True` mode.

Note that `cbreak` overrides `raw`.

The `raw=False` and `cbreak=False` calls follow historical practice in that they attempt to restore to normal ('cooked') mode from `raw` and `cbreak` modes respectively. M

Mixing (`raw` is `True` or `False`) and (`cbreak` is `True` or `False`) calls leads to tty driver control states that are hard to predict or understand; it is not recommended.

Note that return `None` if the property have never been set.

[See `curs_getch(3X)` for a discussion of how these routines interact with `echo=True` and `echo=False`.]

**Returns** The `cbreak` property value

**Return type** `bool`

## **echo**

Control whether characters typed by the user are echoed by `getch` as they are typed.

Echoing by the tty driver is always disabled, but initially `getch` is in echo mode, so characters typed are echoed.

Authors of most interactive programs prefer to do their own echoing in a controlled area of the screen, or not to echo at all, so they disable echoing by calling `noecho`.

[See `curs_getch(3X)` for a discussion of how these routines interact with `cbreak` and `nocbreak`.]

:return the echo property value :rtype: `bool`

## **halfdelay**

The `halfdelay` property is used for half-delay mode, which is similar to `cbreak` mode in that characters typed by the user are immediately available to the program.

However, after blocking for `tenths` tenths of seconds, `ERR` is returned if nothing has been typed.

**Returns**

## **intrflush**

If the `intrflush` property is enabled, (`bf` is `TRUE`), when an interrupt key is pressed on the keyboard (interrupt, break, quit) all output in the tty driver queue will be flushed, giving the effect of faster response to the interrupt, but causing curses to have the wrong idea of what is on the screen.

Disabling (`bf` is `FALSE`), the option prevents the flush.

The default for the option is inherited from the tty driver settings. The `window` argument is ignored.

Note: That return `None` only if property have never been set

**Returns** The `intrflush` property value

**Return type** `bool` or *None*

## **keypad**

The `keypad` property enables the keypad of the user's terminal.

If enabled (`bf` is `TRUE`), the user can press a function key (such as an arrow key) and `wgetch` returns a single value representing the function key, as in `KEY_LEFT`. If disabled (`bf` is `FALSE`), `curses` does not treat function keys specially and the program has to interpret the escape sequences itself.

If the keypad in the terminal can be turned on (made to transmit) and off (made to work locally), turning on this option causes the terminal keypad to be turned on when `wgetch` is called.

The default value for `keypad` is `True`.

Note: That return `None` if `keypad` have never been set.

**Returns** The `keypad` property value

**Return type** `bool` or *None*

**instance** = `<GLXCurses.libs.TTY.Screen object>`

#### **meta**

Initially, whether the terminal returns `def_prog_mode` 7 or 8 significant bits on input depends on the control mode of the tty driver [see `termio(7)`].

To force 8 bits to be returned, invoke `meta`` = ``True` this is equivalent, under POSIX, to setting the CS8 flag on the terminal.

To force 7 bits to be returned, invoke `meta`` = ``False` this is equivalent, under POSIX, to setting the CS7 flag on the terminal.

If the terminfo capabilities `smm` (`meta_on`) and `rmm` (`meta_off`) are defined for the terminal, `smm` is sent to the terminal when `meta`` = ``True` is called and `rmm` is sent when `meta`` = ``False` is called.

Note: That return `None` when the property have never been set

**Returns** The `meta` property value

**Return type** `bool` or *None*

#### **nodelay**

The `nodelay` option causes `getch` to be a non-blocking call. If no input is ready, `getch` returns `ERR`. If disabled (`bf` is `FALSE`), `getch` waits until a key is pressed.

While interpreting an input escape sequence, `wgetch` sets a timer while waiting for the next character. If `notimeout(win, TRUE)` is called, then `wgetch` does not set a timer. The purpose of the timeout is to differentiate between sequences received from a function key and those typed by a user.

**Returns** The `nodelay` property value

**Return type** `bool` or *None*

#### **raw**

The `raw` property place the terminal into or out of raw mode.

Raw mode is similar to `cbreak` mode, in that characters typed are immediately passed through to the user program. The differences are that in raw mode, the interrupt, quit, suspend, and flow control characters are all passed through uninterpreted, instead of generating a signal.

The behavior of the `BREAK` key depends on other bits in the tty driver that are not set by `curses`.

**Returns** The property value

**Return type** `bool` or *None*

#### **qiflush**

When (`qiflush` is `False`) normal flush of input and output queues associated with the `INTR`, `QUIT` and `SUSP` characters will not be done [see `termio(7)`].

When (`qiflush` is `True`) is called, the queues will be flushed when these control characters are read.

You may want use (qiflush is False) in a signal handler if you want output to continue as though the interrupt had not occurred, after the handler exits.

**Returns** The `qiflush` property value

**Return type** bool or *None*

### **timeout**

The `timeout` and `wtimeout` routines set blocking or non-blocking read for a given window.

If delay is negative, blocking read is used (i.e., waits indefinitely for input).

If delay is zero, then non-blocking read is used (i.e., read returns ERR if no input is waiting).

If delay is positive, then read blocks for delay milliseconds, and returns ERR if there is still no input.

Hence, these routines provide the same functionality as `nodelay`, plus the additional capability of being able to block for only delay milliseconds (where delay is positive).

**Returns**

### **close()**

A Application must be close properly for permit to Curses to clean up everything and get back the tty in startup condition

Generally that is follow by a `sys.exit(0)` for generate a exit code.

### **lowlevel\_getch()**

Use by the Mainloop for interact with teh keyboard and the mouse.

`getch()` returns an integer corresponding to the key pressed.

If it is a normal character, the integer value will be equivalent to the character. Otherwise it returns a number which can be matched with the constants defined in `curses.h`.

For example if the user presses F1, the integer returned is 265.

This can be checked using the macro `KEY_F()` defined in `curses.h`.

This makes reading keys portable and easy to manage.

```
ch = GLXCurses.Screen().lowlevel_getch()
```

`lowlevel_getch()` will wait for the user to press a key, (unless you specified a timeout) and when user presses a key, the corresponding integer is returned.

Then you can check the value returned with the constants defined in `curses.h` to match against the keys you want.

```
if ch == curses.KEY_LEFT:
    print("Left arrow is pressed")
```

**Returns** an integer corresponding to the key pressed.

**Return type** int

### **reset\_screen()**

### **refresh()**

### **touch\_screen()**

### **static check\_terminal(force\_xterm=False)**

### **static get\_mouse()**

**class** GLXCurses.Colors

Bases: object

**itu\_recommendation**

Get `itu_recommendation` property value

Where:

<https://en.wikipedia.org/wiki/ITU-R>

[https://en.wikipedia.org/wiki/Rec.\\_601](https://en.wikipedia.org/wiki/Rec._601)

[https://en.wikipedia.org/wiki/Rec.\\_709](https://en.wikipedia.org/wiki/Rec._709)

[https://en.wikipedia.org/wiki/Rec.\\_2100](https://en.wikipedia.org/wiki/Rec._2100)

Allowed Value: 'BT.601', 'BT.709', 'BT.2100' Default Value: 'BT.601'

**Returns** `itu_recommendation` property value

**Return type** str

**color\_detection\_value**

**static** `curses_color` (*color*)

A "translation" function that converts standard-intensity CGA color numbers (0 to 7) to curses color numbers, using the curses constant names like `COLOR_BLUE` or `COLOR_RED`

**Parameters** `color` –

**Returns** `curses.COLOR`

**static** `curses_color_pair_number` (*fg*, *bg*)

A function to set an integer bit pattern based on the classic color byte

**Parameters**

- **fg** (*int*) – Foreground color
- **bg** (*int*) – Background color

**curses\_color\_pairs\_init** ()

It function create all possible color pairs

**Returns**

**static** `strip_hash` (*str\_rgb*)

Strip leading # if exists.

**Parameters** **str\_rgb** (*str*) – the str it contain a # or not

**Returns** a str without #

**Return type** str

**get\_luma\_component\_rgb** (*r*, *g*, *b*)

**static** `rgb_to_ansi16` (*r*, *g*, *b*)

**rgb\_to\_curses\_attributes** (*r*, *g*, *b*)

**rgb\_hex\_to\_list\_int** (*str\_rgb*)

**color** (*fg=None*, *bg=None*, *attributes=None*)

Convert a RGB value to a directly usable curses color

`draw(y, x, "Hello", color)` where the return of it function is directly usable

**Returns** `color.pair` | `curses.Attribut`

**Return type** int

**hex\_rgb\_to\_curses** (*fg=None, bg=None*)

Convert a RGB value to a directly usable curses color

draw(y, x, "Hello", color) where the return of it function is directly usable

bg='#000000', FG='#FFFFFF'

**Returns** color.pair | curses.Attribut

**Return type** int

**class** GLXCurses.Style

Bases: *GLXCurses.libs.Colors.Colors*

### Description

Galaxie Curses Style is equivalent to a skin feature, the entire API receive a common Style from Application and each individual Widget can use it own separate one.

Yet it's a bit hard to explain how create you own Style, in summary it consist to a dict() it have keys with a special name call `Attribute`, inside that dictionary we create a second level of dict() dedicated to store color value of each States

### default\_attributes\_states

Return a default style, that will be use by the entire GLXCurses API via the `__attribute_states` object. every Widget's will receive it style by default.

**Returns** A Galaxie Curses Style dictionary

**Return type** dict

### attributes\_states

Return the `__attribute_states` attribute, it consist to a dictionary it store a second level of dictionary with keys if have special name.

**Returns** attribute states dictionary on Galaxie Curses Style format

**Return type** dict

**attribute\_to\_rgb** (*attribute='base', state='STATE\_NORMAL'*)

Return a text color, for a attribute and a state passed as argument, it's use by widget for know which color use, when a state change.

By example: When color change if the button is pressed

### Parameters

- **attribute** – accepted value: `text_fg`
- **state** – accepted value: `STATE_NORMAL`, `STATE_ACTIVE`, `STATE_PRELIGHT`, `STATE_SELECTED`, `STATE_INSENSITIVE`

**Returns** text color

**Return type** str

**class** GLXCurses.Object

Bases: *glxeveloploop.bus.Bus*

### Description

Object is the fundamental type providing the common attributes and methods for all object types in GLXCurses.

The Object class provides methods for object construction and destruction, property access methods, and signal support.

Signals are described in detail here.

**id**

Return the `id` property value

**Returns** a unique id

**Return type** str

**children****debug****debug\_level**

Get the debugging information's level to display on the stdscr.

Range: 0 to 3

**Returns** The `debug_level` property value

**Return type** int

**flags**

Return the `flags` attribute, it consist to a dictionary it store keys with have special name.

**Returns** a Dictionary with Galaxie Curses Object Flags format

**Return type** dict

**default\_flags****destroy()**

Destroy the object

**eveloop\_dispatch** (*detailed\_signal*, *args*)

Inform every children or child about a event and execute a eventual callback

**Parameters**

- **detailed\_signal** (*str*) – a string containing the signal name
- **args** (*list*) – additional parameters arg1, arg2

**class** GLXCurses.Widget

Bases: `GLXCurses.Object.Object`, `GLXCurses.Aera.Area`, `GLXCurses.libs.Colorable.Colorable`

**Return type** object

**preferred\_height****preferred\_width****app\_paintable**

Whether the application will paint directly on the widget.

Owner: Widget Flags: Read / Write Default value: FALSE

**Returns** True or False

**Return type** bool

**can\_default**

Whether the widget can be the default widget.

Owner: Widget Flags: Read / Write Default value: False



**Returns** True if widget can be a default widget, False otherwise

**Return type** bool

## events

Owner: Widget Flags: Read Default value: False

**Returns** list of flags content in GLXC.EventMask

**Return type** list

## can\_focus

Whether the widget can accept the input focus.

Owner: Widget Flags: Read / Write Default value: False

**Returns** True or False

**Return type** bool

## can\_prelight

If True if the widget will display prelight color.

By default that if False, by exemple a container is a hidden Widget and have no raison to display prelight.

At the oposit the prelight of a button can be disable with it property

**Returns** True or False

**Return type** bool

## composite\_child

Whether the widget is part of a composite widget.

**Returns** True or False

**Return type** bool

## expand

Whether to expand in both directions. Setting this sets both “hexpand” and “vexpand”

**Returns** True or False

**Return type** bool

## focus\_on\_click

Whether the widget should grab focus when it is clicked with the mouse.

This property is only relevant for widgets that can take focus.

**Returns** True or False

**Return type** bool

## halign

How to distribute horizontal space if widget gets extra space, see GLXC.Align

**Allowed value:**

**Stretch to fill all space if possible, center if no meaningful way to stretch** GLXC.ALIGN\_FILL = ‘FILL’

**Snap to left or top side, leaving space on right or bottom** GLXC.ALIGN\_START = ‘START’

**Snap to right or bottom side, leaving space on left or top** GLXC.ALIGN\_END = ‘END’

**Center natural width of widget inside the allocation** GLXC.ALIGN\_CENTER = 'CENTER'

**Align the widget according to the baseline.** GLXC.ALIGN\_BASELINE = 'BASELINE'

**Returns** a GLXC.Align

**Return type** str

**has\_default**

Whether the widget is the default widget.

**Returns** True or False

**Return type** bool

**has\_focus**

Whether the widget has the input focus.

**Returns** True or False

**Return type** bool

**has\_prelight**

Whether the widget is pre light.

**Returns** True or False

**Return type** bool

**has\_tooltip**

Enables or disables the emission of “query-tooltip” on widget . A value of `True` indicates that widget can have a tooltip, in this case the widget will be queried using “query-tooltip” to determine whether it will provide a tooltip or not.

**Returns** True or False

**Return type** bool

**height\_request**

Override for height request of the widget, or -1 if natural request should be used.

**Returns** height\_request property value

**Return type** int

**hexpand**

Whether to expand horizontally.

**Returns** True if the widget have to expand horizontally

**Return type** bool

**hexpand\_set**

Whether to use the “hexpand” property

**Returns** True if the widget use the “hexpand” property

**Return type** bool

**is\_focus**

Whether the widget is the focus widget within the toplevel.

**Returns** True if the widget is the focus widget within the toplevel.

**Return type** bool

**margin**

All four sides' margin at once. If read, returns max margin on any side.

Allowed values: [0,32767]

**Returns** max margin on any side

**Return type** int

**margin\_bottom**

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `Widget.set_size_request()` for example.

Allowed values: [0,32767]

**Returns** Margin on bottom side of widget.

**Return type** int

**margin\_end**

Margin on end of widget, horizontally. This property supports left-to-right and right-to-left text directions.

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `Widget.set_size_request()` for example.

Allowed values: [0,32767]

**Returns** Margin on end of widget, horizontally.

**Return type** int

**margin\_start**

Margin on start of widget, horizontally. This property supports left-to-right and right-to-left text directions.

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `Widget.set_size_request()` for example.

Allowed values: [0,32767]

**Returns** Margin on start of widget, horizontally.

**Return type** int

**margin\_top**

Margin on top side of widget.

This property adds margin outside of the widget's normal size request, the margin will be added in addition to the size from `Widget.set_size_request()` for example.

Allowed values: [0,32767]

**Returns** Margin on top side of widget.

**Return type** int

**name**

The name of the widget.

**Returns** name of the widget.

**Return type** str

**no\_show\_all**

Whether `Widget.show_all()` should not affect this widget.

**Returns** If True, `Widget.show_all()` should not affect this widget

**Return type** bool

**parent**

The parent GLXCurses.Container of this GLXCurses.Widget. Must be a GLXCurses.Container.

**Returns** The parent of the GLXCurses.Widget

**Return type** GLXCurses.Container

**receives\_default**

If True, the widget will receive the default action when it is focused.

**Returns** True if the widget receives default

**Return type** bool

**sensitive**

Whether the widget responds to input.

**Returns** True if the widget responds to input

**Return type** bool

**style**

The style of the widget, which contains information about how it will look (colors, etc).

**Returns** a GLXCurses.Style instance

**Return type** GLXCurses.Style

**tooltip\_text**

This is a convenience property which will take care of getting the tooltip shown if the given string is not NULL: `has-tooltip` will automatically be set to TRUE and there will be taken care of “query-tooltip” in the default signal handler.

**Returns** tooltip\_text property value

**Return type** str or *None*

**valign**

How to distribute vertical space if widget gets extra space, see GLXC.Align

Default value: GLXC.ALIGN\_FILL

**Returns** The “valign” property

**Return type** GLXC.Align

**vexpand**

Whether to expand vertically. See `Widget().set_vexpand()`.

**Returns** True if the widget have to expand vertically

**Return type** bool

**vexpand\_set**

Whether to use the “vexpand” property

**Returns** True if the widget use the “vexpand” property

**Return type** bool

**visible**

Whether the widget is visible.

Default value: False

**Returns** True if the widget is visible.

**Return type** bool

**width\_request**

Override for width request of the widget, or -1 if natural request should be used.

**Returns** width\_request property value

**Return type** int

**window**

The widget's window if it is realized, *None* otherwise.

**Returns** return the Window object if realized, *None* otherwise

**Return type** GLXCurses.Window or *None*

**new()**

Not totally like GTK yet ...

**Actually:** The Widget.New() “can be” and “is” overridden by each GLXCurses Components.

**Original GTK:** This is a convenience function for creating a widget and setting its properties in one go. For example you might write: `Widget().New(GLXC.TYPE_LABEL, “label”, “Hello World”, “xalign”, 0.0, NULL)` to create a left-aligned label.

**Returns** the GLXCurses.Widget

**Return type** GLXCurses.Widget

**destroy()**

Destroy the object

**in\_destruction**

Returns whether the widget is currently being destroyed.

This information can sometimes be used to avoid doing unnecessary work.

**Returns** True if widget is being destroyed

**Return type** bool

**destroyed** (*widget=None, widget\_pointer=None*)**unparent** (*widget=None*)

This function is only for use in widget implementations. Should be called by implementations of the remove method on Container, to dissociate a child from the container.

**Parameters** **widget** (*GLXCurses.Widget*) – a GLXCurses.Widget

**Raises**

- **TypeError** – if *widget* is not a valid GLXCurses type.
- **TypeError** – if *widget* is not a instance of GLXCurses.Widget.

**show()**

Flags a widget to be displayed. Any widget that isn't shown will not appear on the stdscr.

If you want to show all the widgets in a container, it's easier to call `GLXCurses.Widget.show_all()` on the container, instead of individually showing the widgets.

Remember that you have to show the containers containing a widget, in addition to the widget itself, before it will appear onscreen.

When a toplevel container is shown, it is immediately realized and mapped; other shown widgets are realized and mapped when their toplevel container is realized and mapped.

**show\_now()**

Shows a widget.

If the widget is an unmapped toplevel widget (i.e. a `GLXCurses.Window` that has not yet been shown), enter the main loop and wait for the window to actually be mapped.

Be careful; because the main loop is running, anything can happen during this function.

**hide()**

Reverses the effects of `GLXCurses.Widget.show()`, causing the widget to be hidden (invisible to the user).

**show\_all()**

Recursively shows a widget, and any child widgets (if the widget is a container).

**map**

This function is only for use in widget implementations. Causes a widget to be mapped if it isn't already.

**realize**

Creates the `GLXCurses` (windowing system) resources associated with a widget. For example, `widget->window` will be created when a widget is realized. Normally realization happens implicitly; if you show a widget and all its parent containers, then the widget will be realized and mapped automatically.

Realizing a widget requires all the widget's parent widgets to be realized; calling `Widget.realize()` realizes the widget's parents in addition to widget itself. If a widget is not yet inside a toplevel window when you realize it, bad things will happen.

This function is primarily used in widget implementations, and isn't very useful otherwise. Many times when you think you might need it, a better approach is to connect to a signal that will be called after the widget is realized automatically, such as "draw". Or simply `g_signal_connect()` to the "realize" signal.

**Returns** the `realize` property value

**Return type** `bool`

**child\_visible**

The `set_child_visible()` method determines if the widget should be mapped along with its parent.

**Returns**

**get\_toplevel()****set\_decorated()** (*decorated*)**get\_decorated()****refresh()****override\_color()** (*color*)**override\_background\_color()** (*color*)**attribute\_states**

Return the `__attribute_states` attribute, it consist to a dictionary it store a second level of dictionary with keys if have special name.

**Returns** attribute states dictionary on Galaxie Curses Style format

**Return type** `dict`

**has\_window**

Determines whether widget has a `GdkWindow` of its own.

See `GLXCurses.Widget.set_has_window()`.

**Returns** `TRUE` if widget has a window, `FALSE` otherwise

**Return type** bool

**draw()**

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**unchild**(*widget=None*)

**class** GLXCurses.Container

Bases: *GLXCurses.Widget.Widget*

GLXCurses.Container — Base class for widgets which contain other widgets

Description:

A GLXCurses user interface is constructed by nesting widgets inside widgets. Container widgets are the inner nodes in the resulting tree of widgets: they contain other widgets. So, for example, you might have a GLXCurses.Window containing a GLXCurses.Frame containing a GLXCurses.Label. If you wanted an image instead of a textual label inside the frame, you might replace the GLXCurses.Label widget with a GLXCurses.Image widget.

There are two major kinds of container widgets in GLXCurses. Both are subclasses of the abstract GLXCurses.Container base class.

The first type of container widget has a single child widget and derives from GLXCurses.Bin. These containers are decorators, which add some kind of functionality to the child. For example, a GLXCurses.Button makes its child into a clickable button; a GLXCurses.Frame draws a frame around its child and a GLXCurses.Window places its child widget inside a top-level window.

The second type of container can have more than one child; its purpose is to manage layout. This means that these containers assign sizes and positions to their children. For example, a GLXCurses.HBox arranges its children in a horizontal row, and a GLXCurses.Grid arranges the widgets it contains in a two-dimensional grid.

For implementations of GLXCurses.Container the virtual method GLXCurses.Container.forall() is always required, since it's used for drawing and other internal operations on the children. If the GLXCurses.Container implementation expect to have non internal children it's needed to implement both GLXCurses.Container.add() and GLXCurses.Container.remove(). If the GLXCurses.Container implementation has internal children, they should be added widget.set\_parent() on \_\_init\_\_() and removed with widget.unparent() in the GLXCurses.Widget.destroy() implementation. See more about implementing custom widgets at <https://wiki.gnome.org/HowDoI/CustomWidgets>

**border\_width**

Set the border\_width property value

Allowed values: <= 65535

Default value: 0

**Returns** The width of the empty border outside the containers children.

**Return type** int

**child**

Set the child property value

**Returns** Child element

**Return type** GLXCurses.ChildElement or *None*

**resize\_mode**

Set the resize\_mode property value

Default value: GLXC.RESIZE\_PARENT

**Returns** Specify how resize events are handled.

**Return type** str

**add** (*widget=None*)

Adds widget to container .

Typically used for simple containers such as Window, Frame, or Button;

For more complicated layout containers such as Box or Grid, this function will pick default packing parameters that may not be correct.

So consider functions such as `GLXCurses.Box.pack_start()` and `GLXCurses.Grid.attach()` as an alternative to `GLXCurses.Container.add()` in those cases.

A widget may be added to only one container at a time; you (should not) place the same widget inside two different containers.

**Parameters** **widget** (*GLXCurses.Widget*) – a widget to be placed inside container

**Raises** **TypeError** – if widget is not a instance of GLXCurses.Widget

**remove** (*widget=None*)

Removes widget from container .

Widget must be inside container .

Note that container will own a reference to widget , and that this may be the last reference held; so removing a widget from its container can destroy that widget. If you want to use widget again, you need to add a reference to it before removing it from a container, using `g_object_ref()`. If you don't want to use widget again it's usually more efficient to simply destroy it directly using `Widget.destroy()` since this will remove it from the container and help break any circular reference count cycles.

**Parameters** **widget** (*GLXCurses Widget*) – a current child of container

**Raises** **TypeError** – if widget is not a instance of GLXCurses.Widget

**add\_with\_properties** (*widget=None, properties=None*)

Adds widget to container , setting child properties at the same time. See `GLXCurses.Container.add()` and `GLXCurses.Container.child_set()` for more details.

**Parameters**

- **widget** (*GLXCurses.Widget*) – a widget to be placed inside container
- **properties** (*GLXCurses.ChildProperty*) – properties to set

**Raises**

- **TypeError** – if properties is not a GLXCurses.ChildProperty instance
- **TypeError** – if widget is not a instance of GLXCurses.Widget

**get\_resize\_mode** ()

Returns the resize mode for the container.

**Allowed value:**

- `GLXC.RESIZE_PARENT`
- `GLXC.RESIZE_QUEUE`
- `GLXC.RESIZE_IMMEDIATE`

**See also:**

`GLXCurses.Container.set_resize_mode()`.



**Warning:** `GLXCurses.Container.get_resize_mode()` has been deprecated since version 3.12 of GTK+, if will be remove as soon of possible.

**Returns** the current resize mode

**Return type** `GLXCurses.Constants`

**set\_resize\_mode** (*resize\_mode=None*)

Sets the resize mode for the container.

The resize mode of a container determines whether a resize request will be passed to the container's parent, queued for later execution or executed immediately.

**Allowed value:**

- `GLXC.RESIZE_PARENT`
- `GLXC.RESIZE_QUEUE`
- `GLXC.RESIZE_IMMEDIATE`

**See also:**

`GLXCurses.Container.get_resize_mode()`.

**Warning:** `GLXCurses.Container.set_resize_mode()` has been deprecated since version 3.12 of GTK+, if will be remove as soon of possible.

**Parameters** **resize\_mode** (`GLXCurses.Constants`) – the new resize mode

**check\_resize** ()

The `check_resize()` method emits the “check-resize” signal on the container.

**foreachs** (*callback, \*callback\_data*)

Invokes `callback` on each non-internal child of container . See `GLXCurses.Container.forall()` for details on what constitutes an “internal” child. For all practical purposes, this function should iterate over precisely those child widgets that were added to the container by the application with explicit `add()` calls.

Most applications should use `GLXCurses.Container.foreachs()`, rather than `GLXCurses.Container.forall()`.

**Parameters**

- **callback** – a callback.
- **callback\_data** – callback user `__area_data`

**get\_path\_for\_child** (*child=None*)

Returns a newly created widget path representing all the widget hierarchy from the toplevel down to and including `child` .

**Returns** A newly created `WidgetPath`

**forall** (*callback, callback\_data*)

**propagate\_expose** (*child, event*)

**set\_focus\_chain** (*focusable\_widgets*)

**get\_focus\_chain** ()

**unset\_focus\_chain** ()

**set\_reallocate\_redraws** (*needs\_redraws*)

**set\_focus\_child** (*child*)

**get\_focus\_child** ()

**get\_focus\_vadjustment** ()

Retrieves the vertical focus adjustment for the container. See [Container.set\\_focus\\_vadjustment\(\)](#).

**Returns** the vertical focus adjustment, or :py:\_\_area\_data:None if none has been set.

**Return type** [Adjustment\(\)](#) or :py:\_\_area\_data:None

**set\_focus\_vadjustment** (*adjustment=None*)

Hooks up an adjustment to focus handling in a container, so when a child of the container is focused, the adjustment is scrolled to show that widget. This function sets the vertical alignment. See [scrolled\\_window\\_get\\_vadjustment\(\)](#) for a typical way of obtaining the adjustment and [Container.set\\_focus\\_hadjustment\(\)](#) for setting the horizontal adjustment.

The adjustments have to be in character units and in the same coordinate system as the allocation for immediate children of the container.

**Parameters** **adjustment** ([Adjustment\(\)](#) or :py:\_\_area\_data:None) – an adjustment which should be adjusted when the focus is moved among the descendants of container

**Raises** **TypeError** – if adjustment is not a [Adjustment\(\)](#)

**get\_focus\_hadjustment** ()

Retrieves the horizontal focus adjustment for the container. See [Container.set\\_focus\\_hadjustment\(\)](#).

**Returns** the horizontal focus adjustment, or :py:\_\_area\_data:None if none has been set.

**Return type** [Adjustment\(\)](#) or :py:\_\_area\_data:None

**set\_focus\_hadjustment** (*adjustment*)

Hooks up an adjustment to focus handling in a container, so when a child of the container is focused, the adjustment is scrolled to show that widget. This function sets the horizontal alignment. See [scrolled\\_window\\_get\\_hadjustment\(\)](#) for a typical way of obtaining the adjustment and [Container.set\\_focus\\_vadjustment\(\)](#) for setting the vertical adjustment.

The adjustments have to be in pixel units and in the same coordinate system as the allocation for immediate children of the container.

**Parameters** **adjustment** ([Adjustment\(\)](#) or :py:\_\_area\_data:None) – an adjustment which should be adjusted when the focus is moved among the descendants of container

**Raises** **TypeError** – if adjustment is not a [Adjustment\(\)](#)

**child\_type** (*container*)

Returns the type of the children supported by the container.

Note that this may return None to indicate that no more children can be added, e.g. for a Paned which already has two children.

Note that this may return -1 to indicate container is not found

**Parameters** **container** –

**Returns** the type of children

**Return type** str, None or -1

**Raises** **TypeError** – if child is not a GLXCurses type as tested by [glxc\\_type\(\)](#)

**child\_set** (*child*, *properties=None*)

Sets one or more child properties for child and container .

**Parameters**

- **child** (A *GLXCurses.Widget*) – a GLXCurses.Widget which is a child of container
- **properties** (*GLXCurses.ChildProperty*) – properties to set

**Raises**

- **TypeError** – if *child* is not a GLXCurses type as tested by `glxc_type()`
- **TypeError** – if *properties* is not a dict type

**child\_get** (*child*)

Gets the values of one or more child properties for child and container .

**Parameters** **child** (A *GLXCurses object*) – a widget which is a child of container

**Returns** properties of the child or None if child not found

**Return type** dict or *None*

**Raises** **TypeError** – if *child* is not a GLXCurses type as tested by `glxc_type()`

**child\_set\_property** (*child*, *property\_name=None*, *value=None*)

Sets a child property for child and container .

**Parameters**

- **child** (a *GLXCurses.Widget*) – a GLXCurses.Widget which is a child of GLXCurses.Container
- **property\_name** (*str*) – the name of the property to set
- **value** (*everything except None*) – the value to set the property to

**Raises**

- **TypeError** – if *child* is not a GLXCurses type as tested by `glxc_type()`
- **TypeError** – if *property\_name* is not str type
- **TypeError** – if *value* is None type

**child\_get\_property** (*child*, *property\_name=None*)

Gets the value of a child property for child and container .

**Parameters**

- **child** (a *GLXCurses Object*) – a widget which is a child of container
- **property\_name** (*str*) – the name of the property to set

**Raises**

- **TypeError** – if *child* is not a GLXCurses type as tested by `glxc_type()`
- **TypeError** – if *property\_name* is not str type

**get\_border\_width** ()

Retrieves the border width of the container.

See GLXCurses.Container.set\_border\_width().

**Returns** the current border width

**Return type** int

**set\_border\_width** (*border\_width=0*)

Sets the border width of the container.

The border width of a container is the amount of space to leave around the outside of the container. The only exception to this is `GLXCurses.Window`; because toplevel windows can't leave space outside, they leave the space inside. The border is added on all sides of the container. To add space to only one side, use a specific "margin" property on the child widget, for example "margin-top".

`border_width` have valid values are in the range 0-65535 chars and will be clamp to value.

**Parameters** `border_width` (*int*) – amount of blank space to leave outside the container.

**Raises** `TypeError` – When `border_width` is not a int

**class** `GLXCurses.TextTag`

Bases: `GLXCurses.Object.Object`

You may wish to begin by reading the text widget conceptual overview which gives an overview of all the objects and data types related to the text widget and how they work together.

Tags should be in the `TextTagTable` for a given `TextBuffer` before using them with that buffer.

For each property of `TextTag`, there is a "set" property, e.g. "font-set" corresponds to "font". These "set" properties reflect whether a property has been set or not.

They are maintained by `GLXCurses` and you should not set them independently.

**accumulative\_margin**

Whether the margins accumulate or override each other.

When set to `True` the margins of this tag are added to the margins of any other non-accumulative margins present.

When set to `False` the margins override one another (the default).

Default value is `False` and be restore when `accumulative_margin` is set to `None`

**Returns** If `True` the margins of this tag are added to the margins of any other non-accumulative

**Return type** `bool`

**background**

Background color as a string.

**Returns** color as a string

**Return type** `str`

**background\_full\_height**

Whether the background color fills the entire line height or only the height of the tagged characters.

When set to `True` the background color fills the entire line height

Default value is `False` and be restore when `background_full_height` is set to `None`

**Returns** If `True` the background color fills the entire line height

**Return type** `bool`

**background\_full\_height\_set**

Whether this tag affects background height.

When set to `True` this tag affects background height

Default value is `False` and be restore when `background_full_height_set` is set to `None`

**Returns** `True` If this tag affects background height

**Return type** bool

### **background\_rgb**

Background color as a RGB.

Default value is {'r': 0, 'g': 0, 'b': 255} and be restore when background\_rgb is set to None

**Returns** The RGB color as dict with **r**, **g**, **b** key\_name

**Return type** dict

### **background\_set**

Whether this tag affects the background color.

Default value is False and be restore when background\_set is set to None

**Returns** If True, this tag affects the background color

**Return type** bool

### **direction**

Text direction, e.g. right-to-left -> 'RTL' or left-to-right -> 'LTR'.

**Returns** GLXC.TextDirection direction type

**Return type** str

### **editable**

Whether the text can be modified by the user.

Default value is True and be restore when editable is set to None

**Returns** If True, the text can be modified by the user.

**Return type** bool

### **editable\_set**

Whether this tag affects text editability.

Default value is False and be restore when editable\_set is set to None

**Returns** If False, text editability is disable

**Return type** bool

### **family**

Name of the font family, e.g. Sans, Helvetica, Times, Monospace.

Default value is None and be restore when family is set to None

**Returns** The font family name

**Return type** str or *None*

### **family\_set**

Whether this tag affects the font family.

Default value is False and be restore when family\_set is set to None

**Returns** If False, text editability is disable

**Return type** bool

## **class** GLXCurses.TextTagTable

Bases: *GLXCurses.Object.Object*

**new()**

Creates a new GLXCurses.TextTagTable. The table contains no tags by default.

**Returns** a new GLXCurses.TextTagTable

**Return type** GLXCurses.TextTagTable

**add()****remove()****lookup()****foreach()****get\_size()****class** GLXCurses.TextBuffer

Bases: *GLXCurses.Object.Object*

**cursor\_position**

The position of the insert mark (as offset from the beginning of the buffer). It is useful for getting notified when the cursor moves.

**Returns** The cursor position

**Return type** int

**has\_selection**

Whether the buffer has some text currently selected.

**Returns** True when buffer have selection

**Return type** bool

**text****class** GLXCurses.TextView

Bases: *GLXCurses.Container.Container*

**accept\_tab**

Whether Tab will result in a tab character being entered.

Default value is True, and be restored when accept\_tab is set to None

**Returns** If True Tab key will produce Tab char

**Return type** bool

**bottom\_margin**

The bottom margin for text in the text view.

**Returns** The bottom margin padding

**Return type** int

**buffer**

The buffer which is displayed.

**Returns** a buffer

**Return type** GLXCurses.TextBuffer

**cursor\_visible**

If the insertion cursor is shown.

Default value is True, and be restored when cursor\_visible is set to None

**Returns** If `True` the cursor will be visible

**Return type** `bool`

**editable**

Whether the text can be modified by the user.

Default value is `True`, and be restored when `editable` is set to `None`

**Returns** If `True` the text can be modified

**Return type** `bool`

**indent**

Amount to indent the paragraph, in chars.

**Returns** indentation in chars

**Return type** `int`

**input\_hints**

Additional hints (beyond “`input_purpose`”) that allow input methods to fine-tune their behaviour.

**Returns** The right margin padding

**Return type** `int`

**input\_purpose**

The purpose of this text field.

This property can be used by on-screen keyboards and other input methods to adjust their behaviour.

**Returns** The right margin padding

**Return type** `int`

**justification**

Left, right, or center justification.

**Returns** `str`

**Return type** `GLXCurses.GLXC.Justification`

**left\_margin**

The left margin for text in the text view.

**Returns** The left margin padding

**Return type** `int`

**overwrite**

Whether entered text overwrites existing contents.

Default value is `False`, and be restored when `overwrite` is set to `None`

**Returns** If `True` text overwrites existing contents

**Return type** `bool`

**populate\_all**

**right\_margin**

The right margin for text in the text view.

**Returns** The right margin padding

**Return type** `int`

**top\_margin**

The top margin for text in the text view.

**Returns** The top margin padding

**Return type** int

**wrap\_mode****class** GLXCurses.Bin

Bases: *GLXCurses.Container.Container*

A container with just one child

**Description**

The *Bin* widget is a container with just one child. It is not very useful itself, but it is useful for deriving subclasses, since it provides common code needed for handling a single child widget.

Many GLXCurses widgets are subclasses of *Bin*, including

- *Window*
- *Button*
- *Frame*
- *HandleBox*
- *ScrolledWindow*

**get\_child()**

Gets the child of the GLXCurses.Bin, or None if the bin contains no child widget.

The returned widget does not have a reference added, so you do not need to unref it.

**Returns** the child of GLXCurses.Bin , or None if it does not have a child.

**Return type** GLXCurses.Bin or *None*

**class** GLXCurses.Box

Bases: *GLXCurses.Container.Container*

**Description**

The *Box* widget organizes child widgets into a rectangular area.

**baseline\_position**

Gets the `baseline_position` value.

**Returns** a GLXC.BaselinePosition

**Return type** GLXC.BaselinePosition

**homogeneous**

Returns whether the *Box* is homogeneous (all children's have the same size).

**See also:**

`Box.set_homogeneous()`

**Returns** True if the *Box* is homogeneous.

**Return type** bool

**spacing**



**new** (*orientation='HORIZONTAL', spacing=None*)  
Creates a new *Box*.

#### Parameters

- **orientation** (*Orientation*) – the box's orientation. Default: `ORIENTATION_HORIZONTAL`
- **spacing** (*int or None*) – the number of characters to place by default between children. Default: 0

**Returns** a new *Box*.

#### Raises

- **TypeError** – if *orientation* is not `glxc.ORIENTATION_HORIZONTAL` or `glxc.ORIENTATION_VERTICAL`
- **TypeError** – if *spacing* is not `int` type or `None`

**pack\_start** (*child=None, expand=True, fill=True, padding=None*)  
Adds child to *Box*, packed with reference to the start of *Box*.

#### Parameters

- **child** (*a GLXCurses Object*) – the widget to be added to *Box*
- **expand** (*bool*) – True if the new child is to be given extra space allocated to *Box* `<GLXCurses.Box.Box>`. The extra space will be divided evenly between all children that use this option
- **fill** (*bool*) – True if space given to child by the *expand* option is actually allocated to child, rather than just padding it. This parameter has no effect if *expand* is set to `False`. A child is always allocated the full height of a horizontal *Box* and the full width of a vertical *Box*. This option affects the other dimension.
- **padding** (*int or None*) – extra space in characters to put between this child and its neighbors, over and above the global amount specified by *spacing* attribute. If child is a widget at one of the reference ends of box, then padding pixels are also put between child and the reference edge of box

#### Raises

- **TypeError** – if *child* is not a GLXCurses type as tested by `glxc_type()`
- **TypeError** – if *expand* is not `bool` type
- **TypeError** – if *fill* is not `bool` type
- **TypeError** – if *padding* is not `int` or `None`

**pack\_end** (*child=None, expand=True, fill=True, padding=None*)  
Adds child to GLXCurses.Box once to the end of GLXCurses.Box.

#### Parameters

- **child** (*GLXCurses.Widget*) – the widget to be added to GLXCurses.Box
- **expand** (*bool*) – True if the new child is to be given extra space allocated to GLXCurses.Box. The extra space will be divided evenly between all children that use this option
- **fill** (*bool*) – True if space given to child by the *expand* option is actually allocated to child, rather than just padding it. This parameter has no effect if *expand* is set to

False. A child is always allocated the full height of a horizontal *Box* and the full width of a vertical *Box*. This option affects the other dimension.

- **padding** (*int* or *None*) – extra space in characters to put between this child and its neighbors, over and above the global amount specified by *spacing* attribute. If child is a widget at one of the reference ends of box , then padding pixels are also put between child and the reference edge of box

#### Raises

- **TypeError** – if *child* is not a instance of *LXCurses.Widget*
- **TypeError** – if *expand* is not bool type
- **TypeError** – if *fill* is not bool type
- **TypeError** – if *padding* is not int or None

#### **reorder\_child** (*child*, *position*)

Moves *child* to a new *position* in the list of *Box* children. The list contains widgets packed *PACK\_START* as well as widgets packed *PACK\_END*, in the order that these widgets were added to *Box*.

A widget's position in the *Box* children list determines where the widget is packed into *Box*. A child widget at some position in the list will be packed just after all other widgets of the same packing type that appear earlier in the list.

#### Parameters

- **child** (*Widget*) – the widget to move
- **position** (*int*) – the new position for *child* in the list of children of *Box*, starting from 0. If negative, indicates the end of the list.

#### Raises

- **TypeError** – if *child* is not a *GLXCurses* type as tested by *glxc\_type()*
- **TypeError** – if *position* is not int type
- **TypeError** – if *child* is not a *GLXCurses* type as tested by *glxc\_type()*

#### **query\_child\_packing** (*child*)

Obtains information about how *child* is packed into box or None if *child* is not found

**Return Key's:** *widget*: the *Widget* of the child to query *expand*: *expand* child property. *fill*: *fill* child property *padding*: *padding* child property. *pack\_type*: *pack-type* child property

**Parameters** *child* (*a Galaxie Widget*) – the *Widget* of to query

**Returns** information about how *child* is packed into box

**Return type** dict or *None*

**Raises** **TypeError** – if *child* is not a *GLXCurses* type as tested by *glxc\_type()*

#### **set\_child\_packing** (*child*, *expand*, *fill*, *padding*, *pack\_type*)

Sets the way *child* is packed into box .

#### Parameters

- **child** (*Widget*) – the *Widget* of the child to set
- **expand** (*bool*) – the new value of the *expand* child property
- **fill** (*bool*) – the new value of the *fill* child property

- **padding** (*int*) – the new value of the padding child property
- **pack\_type** (*PackType*) – the new value of the pack-type child property

**Raises**

- **TypeError** – if child is not bool type
- **TypeError** – if expand is not bool type
- **TypeError** – if padding is not int or None
- **TypeError** – if pack\_type is not glxc.PACK\_START or glxc.PACK\_END

**set\_center\_widget** (*widget=None*)

Sets a center widget; that is a child widget that will be centered with respect to the full width of the box, even if the children at either side take up different amounts of space.

**Parameters** *widget* (*Widget* or None) – the *Widget* of the child to set

**Raises** **TypeError** – if *widget* is not a GLXCurses type as tested by `glxc_type()` or None

**get\_center\_widget** ()

Retrieves the center widget of the box.

**Returns** the center widget or None in case no center widget is set.

**class** GLXCurses.VBox

Bases: *GLXCurses.Box.Box*, *GLXCurses.libs.Dividable.Dividable*

**new** (*homogeneous=True*, *spacing=None*)

Creates a new GLXCurses *VBox*

**Parameters**

- **homogeneous** (*bool*) – True if all children are to be given equal space allotments.
- **spacing** (*int*) – The number of characters to place by default between children.

**Returns** a new *VBox*.

**Raises**

- **TypeError** – if *homogeneous* is not bool type
- **TypeError** – if *spacing* is not int type or None

**draw\_widget\_in\_area** ()

Be here for be overwrite by every widget

**update\_preferred\_sizes** ()

**class** GLXCurses.HBox

Bases: *GLXCurses.Box.Box*, *GLXCurses.libs.Dividable.Dividable*

**Description**

The *HBox* is a container that organizes child widgets into a single row.

Use the *Box* packing interface to determine the arrangement, spacing, width, and alignment of *HBox* children.

All children are allocated the same height.

**new** (*homogeneous=True*, *spacing=None*)

Creates a new GLXCurses *HBox*

**Parameters**

- **homogeneous** (*bool*) – True if all children are to be given equal space allotments.
- **spacing** (*int*) – The number of characters to place by default between children.

**Returns** a new *HBox*.

**Raises**

- **TypeError** – if *homogeneous* is not bool type
- **TypeError** – if *spacing* is not int type or None

**draw\_widget\_in\_area** ()

Be here for be overwrite by every widget

**update\_preferred\_sizes** ()

**class** GLXCurses.Window (*window\_type*= 'TOPLEVEL')

Bases: *GLXCurses.Bin.Bin*

Creates a new *Window*, which is a toplevel window that can contain other widgets.

Nearly always, the type of the window should be GLXC.WINDOW\_TOPLEVEL.

If you're implementing something like a popup menu from scratch (which is a bad idea, just use *Menu*), you might use GLXC.WINDOW\_POPUP. GLXC.WINDOW\_POPUP is not for dialogs, though in some other toolkits dialogs are called "popups".

If you simply want an undecorated window (no window borders), use *decorated* property, don't use GLXC.WINDOW\_POPUP.

**Parameters** *window\_type* (*str*) – type of window contain on GLXC.WindowType list

**Returns** a new *Window*.

**Return type** *Window*

**Raises** **TypeError** – if *window\_type* is not in valid GLXC.WindowType list

**accept\_focus**

Whether the window should receive the input focus.

Default value: TRUE

**Returns** the *accept\_focus* property value

**Return type** bool

**Raises** **TypeError** – When *accept\_focus* is not a bool type

**application**

The *Application* associated with the window.

The application will be kept alive for at least as long as it has any windows associated with it (see *application\_hold()* for a way to keep it alive without windows).

Normally, the connection between the application and the window will remain until the window is destroyed, but you can explicitly remove it by setting the *:application* property to NULL.

**Returns** The *Application* associated with the window or None

**Return type** *GLXCurses.Application.Application* or *None*

**Raises** **TypeError** – When *application* property value is not a *GLXCurses.Application* instance

**attached\_to**

The widget to which this window is attached. See `GLXCurses.Window().set_attached_to()`.

Examples of places where specifying this relation is useful are for instance a Menu created by a `ComboBox`, a completion popup window created by `Entry` or a typeahead search entry created by `TreeView`.

**Returns** The `attached_to` property value

**Return type** `GLXCurses.Widget` or *None*

**decorated**

Whether the window should be decorated by the window manager.

Default is `True` :return: the `decorated` property value :rtype: `bool`

**default\_height**

The default height of the window, used when initially showing the window.

**Returns** the `default_height` property value

**Return type** `int`

**default\_width**

The default width of the window, used when initially showing the window.

**Returns** the `default_width` property value

**Return type** `int`

**deletable**

Whether the window frame should have a close button.

**Returns** The `deletable` property value

**Return type** `bool`

**destroy\_with\_parent**

Get the `destroy_with_parent` property value

**Returns** `True` if the window will be destroyed when the parent is destroyed.

**Return type** `bool`

**focus\_on\_map**

Whether the window should receive the input focus when mapped.

**Returns** the `focus_on_map` property value

**Return type** `bool`

**focus\_visible**

Whether ‘focus rectangles’ are currently visible in this window.

**Returns** The `focus_visible` property value

**Return type** `bool`

**gravity**

Window gravity defines the meaning of coordinates passed to `Window.move()`.

See `Window.move()` for more details.

The default window gravity is `GLXC.GRAVITY_NORTH_WEST` which will typically “do what you mean.”

**Returns** window gravity

**Return type** `str`

**has\_resize\_grip**

Whether the window has a corner resize grip.

Note that the resize grip is only shown if the window is actually resizable and not maximized.

Use “resize-grip-visible” to find out if the resize grip is currently shown.

**Returns** The `has_resize_grip` property value

**Return type** `bool`

**has\_toplevel\_focus**

Whether the input focus is within this Window.

**Returns** The `has_toplevel_focus` property value

**Return type** `bool`

**hide\_titlebar\_when\_maximized**

Whether the titlebar should be hidden during maximization.

**Returns** The `hide_titlebar_when_maximized` property value

**Return type** `bool`

**icon**

Icon for this window.

**Returns** The `icon` property value

**Return type** `curses` Extended Characters

**icon\_name**

The `icon_name` property specifies the name of the themed icon to use as the window icon.

See `IconTheme` for more details.

**Returns** The `icon_name` property value

**Return type** `str` or *None*

**is\_active**

Whether the toplevel is the current active window.

**Returns** The `is_active` property value

**Return type** `bool`

**is\_maximized**

Whether the window is maximized.

**Returns** The `is_maximized` property value

**Return type** `bool`

**mnemonics\_visible**

Whether mnemonics are currently visible in this window.

This property is maintained by `GLXCurses` based on user input, and should not be set by applications.

**Returns** The `mnemonics_visible` property value

**Return type** `bool`

**modal**

If *True*, the window is modal (other windows are not usable while this one is up).

**Returns** The `modal` property value

**Return type** bool

### **resizable**

Gets the value set to `resizable` property.

**Returns** True if the user can resize the window

**Return type** bool

### **role**

Unique identifier for the window to be used when restoring a session.

**Returns** A unique identifier

**Return type** str or *None*

### **screen**

The screen where this window will be displayed.

**Returns** The screen where this window will be displayed

**Return type** *GLXCurses.Screen* or *None*

### **skip\_pager\_hint**

True if the window should not be in the pager.

**Returns** The `skip_pager_hint` property value

**Return type** bool

### **skip\_taskbar\_hint**

True if the window should not be in the task bar.

**Returns** The `skip_taskbar_hint` property value

**Return type** bool

### **startup\_id**

The `startup_id` was originally write-only property for setting window's startup notification identifier.

See `Window.set_startup_id()` for more details.

**Returns** A identifier or None

**Return type** str or *None*

### **title**

The title of the window.

Default value: None

**Returns** the `title` property value

**Return type** str or *None*

### **transient\_for**

Fetches the transient parent for this `GLXCurses.Window`.

See `transient_for.setter` for more details about transient windows.

**Returns** the transient parent for this `GLXCurses.Window`, or None if no transient parent has been set.

**Return type** `GLXCurses.Window` or *None*

### **type**

Return the `type` property

**Returns** GLXC.WindowType

**Return type** str

**type\_hint**

Hint to help the desktop environment understand what kind of window this is and how to treat it.

These are hints for the window manager that indicate what type of function the window has.

The window manager can use this when determining decoration and behaviour of the window.

The hint must be set before mapping the window.

**Returns** hint for the window manager

**Return type** str

**urgency\_hint**

True if the window should be brought to the user's attention.

**Returns** the urgency\_hint property value

**Return type** bool

**position**

The initial position of the window.

**Returns** position constraint

**Return type** str

**decoration\_button\_layout**

Decorated button layout property

**Returns** a layout

**Return type** str

**decoration\_resize\_handle**

The decoration\_resize\_handle property

**Returns** Decoration resize handle size.

**Return type** int

**color**

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**static add\_accel\_group()**

Not implemented

**Raises** **NotImplementedError** – because AccelGroup is not implemented

**static remove\_accel\_group()**

Not implemented

**Raises** **NotImplementedError** – because AccelGroup is not implemented

**activate\_focus()**

Activates the current focused widget within the window.

**Returns** True if a widget got activated.

**Return type** bool



**activate\_default()**

Activates the default widget for the window, unless the current focused widget has been configured to receive the default action (see `gtk_widget_set_receives_default()`), in which case the focused widget is activated.

**Returns****get\_focus()**

The `get_focus()` method returns the current focused widget within the window.

The focus widget is the widget that would have the focus if the toplevel window is focused.

**Returns** The current focused `GLXCurses.Widget`

**Return type** `GLXCurses.Widget` or *None*

**set\_default (default\_widget=None)**

The default widget is the widget that's activated when the user presses Enter in a dialog (for example). This function sets or unsets the default widget for a *Window*. When setting (rather than unsetting) the default widget it's generally easier to call `Widget.grab_default()` on the widget. Before making a widget the default widget, you must call `Widget.set_can_default()` on the widget you'd like to make the default.

**Parameters** **default\_widget** (`GLXCurses.Window`) – a `GLXCurses.Window` or *None* of unset

**Raises** **TypeError** – if `default_widget` is not a `GLXCurses.Widget` instance .

**get\_default\_widget()**

Returns the default widget for window . `GLXCurses.Window().set_default()` for more details.

**Returns** the default `GLXCurses.Widget`, or *None* if there is none.

**Return type** `GLXCurses.Widget` or *None*

**get\_window\_type()**

Gets the type of the window.

Constants.`GLXC.WindowType` are `GLXC.WINDOW_TOPLEVEL` and `GLXC.WINDOW_POPUP`

**Returns** the type of the window

**Return type** `GLXC.WINDOW_TOPLEVEL` or `GLXC.WINDOW_POPUP`

**update\_preferred\_sizes()****class GLXCurses.RadioButton**

Bases: `GLXCurses.Widget.Widget`, `GLXCurses.libs.Movable.Movable`

**active**

**text**

**interface**

**color**

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**update\_preferred\_sizes()****class GLXCurses.CheckButton**

Bases: `GLXCurses.Widget.Widget`, `GLXCurses.libs.Movable.Movable`

**active**

**text**

**interface**

**color**

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**update\_preferred\_sizes()**

**class** GLXCurses.**Adjustment**

Bases: *GLXCurses.Object.Object*

A representation of an adjustable bounded value

**Properties**

**lower**

The minimum value of the adjustment.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**page\_increment**

The page increment of the adjustment.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**page\_size**

The page size of the adjustment. Note that the page-size is irrelevant and should be set to zero if the adjustment is used for a simple scalar value, e.g. in a `SpinButton`.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**step\_increment**

The step increment of the adjustment.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**minimum\_increment**

The smaller of step increment and page increment.

**Type** float

**Flags** Read / Write

**Default value** 0.0

**upper**

The maximum value of the adjustment.

**Type** float

**Flags** Read / Write**Default value** 0.0

---

**Note:** The values will be restricted by `upper - page-size` if the `page-size` property is nonzero.

---

**value**

The value of the adjustment.

**Type** float**Flags** Read / Write**Default value** 0.0**Description**

The *Adjustment* object represents a value which has an associated lower and upper bound, together with step and page increments, and a page size. It is used within several widgets, including *SpinButton*, *Viewport*, and *Range* (which is a base class for *Scrollbar* and *Scale*).

The Adjustment object does not update the value itself. Instead it is left up to the owner of the *Adjustment* to control the value.

**Functions**

**new** (*value=0.0, lower=0.0, upper=0.0, step\_increment=0.0, page\_increment=0.0, page\_size=0.0*)

Creates a new *GLXCurses.Adjustment*.**Parameters**

- **value** (*float*) – The initial value
- **lower** (*float*) – The minimum value
- **upper** (*float*) – The maximum value
- **step\_increment** (*float*) – The step increment
- **page\_increment** (*float*) – The page increment
- **page\_size** (*float*) – The page size

**Returns** a new *GLXCurses.Adjustment***Return type** *GLXCurses.Adjustment***Raises**

- **TypeError** – if *value* is not float
- **TypeError** – if *lower* is not float
- **TypeError** – if *upper* is not float
- **TypeError** – if *step\_increment* is not float
- **TypeError** – if *page\_increment* is not float
- **TypeError** – if *page\_size* is not float

**get\_value** ()Gets the current value of the adjustment. See *set\_value*()**Returns** A current value Adjustment**Return type** float

**set\_value** (*value*)

Set the *Adjustment value* attribute.

The value passed as argument is clamped to lie between *lower* and *upper* attributes.

---

**Note:** For adjustments which are used in a *Scrollbar*, the effective range of allowed values goes from *lower* to *upper - page\_size*.

---

**Raises `TypeError`** – when *value* passed as argument is not a `:py:__area_data:float`

**clamp\_page** (*lower=None, upper=None*)

Updates the *value* attribute to ensure that the range between *lower* and *upper* parameters is in the current page (i.e. between *value* and *value + page\_size*).

If the range is larger than the page size, then only the start of it will be in the current page. A **value-changed** signal will be emitted if the value is changed.

#### Parameters

- **lower** (*float*) – the lower value
- **upper** (*float*) – the upper value

#### Raises

- **TypeError** – when *lower* are not `:py:__area_data:float` type
- **TypeError** – when *upper* are not `:py:__area_data:float` type

**emit\_changed** ()

Emits a “changed” signal from the *Adjustment*.

This is typically called by the owner of the *Adjustment*, after it has changed any of the *Adjustment* attributes other than the value.

**emit\_value\_changed** ()

Emits a “value-changed” signal from the *Adjustment*. This is typically called by the owner of the *Adjustment* after it has changed the “value” property.

**configure** (*value=None, lower=None, upper=None, step\_increment=None, page\_increment=None, page\_size=None*)

Sets all properties of the adjustment at once.

Use this function to avoid multiple emissions of the “changed” signal.

See *Adjustment.set\_lower()* for an alternative way of compressing multiple emissions of “changed” into one.

#### Parameters

- **value** (*float*) – the new value
- **lower** (*float*) – the new minimum value
- **upper** (*float*) – the new maximum value
- **step\_increment** (*float*) – the new step increment
- **page\_increment** (*float*) – the new page increment
- **page\_size** (*float*) – the new page size

**Raises `TypeError`** – when one of parameters are not `:py:__area_data:float` type

**get\_lower()**

Retrieves the minimum value of the adjustment.

**Returns** The current minimum value of the adjustment

**Return type** float

**get\_page\_increment()**

Retrieves the page increment of the adjustment.

**Returns** The current page increment of the adjustment

**Return type** float

**get\_page\_size()**

Retrieves the page size of the adjustment.

**Returns** The current page size of the adjustment

**Return type** float

**get\_step\_increment()**

Retrieves the step increment of the adjustment.

**Returns** The current step increment of the adjustment.

**Return type** float

**get\_minimum\_increment()**

Get the smaller of step increment and page increment. Note that value is compute, then it have no need of a `set_minimum_increment()` method.

**Returns** the minimum increment of adjustment

**Return type** float

**get\_upper()**

Retrieves the maximum value of the adjustment.

**Returns** The current maximum value of the adjustment

**Return type** float

**set\_lower(lower)**

Sets the minimum value of the adjustment.

When setting multiple adjustment properties via their individual setters, multiple `Adjustment.changed()` signals will be emitted. However, since the emission of the `Adjustment.changed()` signal is tied to the emission of the `notify` signals of the changed properties, it's possible to compress the `Adjustment.changed()` signals into one by calling `object_freeze_notify()` and `object_thaw_notify()` around the calls to the individual setters.

Alternatively, using `Adjustment.configure()` has the same effect of compressing `Adjustment.changed()` emissions.

**Warning:** Unfortunately `object_freeze_notify()` and `object_thaw_notify()` don't exist yet. then only `Adjustment.configure()` will make the work.

**Parameters** `lower` (*float*) – the new minimum value

**Raises** **TypeError** – when “lower” argument is not a `:py:__area_data:float`

**set\_page\_increment** (*page\_increment*)  
Sets the page increment of the adjustment.

**See also:**

*Adjustment.set\_lower()* about how to compress multiple emissions of the *Adjustment.changed()* signal when setting multiple adjustment attributes.

**Parameters** *page\_increment* (*float*) – the new page increment

**Raises** **TypeError** – when “page\_increment” argument is not a *:py:\_\_area\_data:float*

**set\_page\_size** (*page\_size*)  
Sets the page size of the adjustment.

**See also:**

*Adjustment.set\_lower()* about how to compress multiple emissions of the *Adjustment.changed()* signal when setting multiple adjustment attributes.

**Parameters** *page\_size* (*float*) – the new page size

**Raises** **TypeError** – when “page\_size” argument is not a *:py:\_\_area\_data:float*

**set\_step\_increment** (*step\_increment*)  
Sets the step increment of the adjustment.

**See also:**

*Adjustment.set\_lower()* about how to compress multiple emissions of the *Adjustment.changed()* signal when setting multiple adjustment attributes.

**Parameters** *step\_increment* (*float*) – the new step increment

**Raises** **TypeError** – when “step\_increment” argument is not a *:py:\_\_area\_data:float*

**set\_upper** (*upper*)  
Sets the maximum value of the adjustment.

**See also:**

*Adjustment.set\_lower()* about how to compress multiple emissions of the *Adjustment.changed()* signal when setting multiple adjustment attributes.

**Parameters** *upper* (*float*) – the new maximum value

**Raises** **TypeError** – when “upper” argument is not a *:py:\_\_area\_data:float*

**class** GLXCurses.Dialog

Bases: *GLXCurses.Window.Window*, *GLXCurses.libs.Movable.Movable*

**action\_aera\_border**

The default border width used around the action area of the dialog, as returned by *Dialog.get\_action\_area()*, unless *GLXCurses.Container.set\_border\_width()* was called on that widget directly.

**Returns** the *action\_aera\_border* property value

**Return type** *int*

**button\_spacing**

Spacing between buttons in Chars

**Returns** the `button_spacing` property value

**Return type** `int`

#### **content\_area\_border**

The default border width used around the content area of the dialog, as returned by `dialog_get_content_area()`, unless `container_set_border_width()` was called on that widget directly.

**Returns** the `content_area_border` property value

**Return type** `int`

#### **content\_area\_spacing**

The default spacing used between elements of the content area of the dialog, as returned by `dialog_get_content_area()`, unless `box_set_spacing()` was called on that widget directly.

**Returns** the `content_area_spacing` property value

**Return type** `int`

#### **new\_with\_buttons** (*title, parent, \*flags*)

##### **Parameters**

- **title** (*str* or `None`) – Title of the dialog, or `None`
- **parent** (*GLXCurses Parent* or `None`) – Transient parent of the dialog, or `None`
- **flags** (*argv*) –

#### **run** ()

Inform Application about the `GLXCurses.Dialog` is active.

Cause Application to forward event only inside the `GLXCurses.Dialog`.

The `GLXCurses.Mainloop` and `GLXCurses.Application` will work with the dialog like a normal `GLXCurses.Window` because dialog is a subclass of `GLXCurses.Window`.

#### **response** (*response\_id=None*)

Emits the “response” signal with the given response ID.

Used to indicate that the user has responded to the dialog in some way; typically either you or `Dialog().run()` will be monitoring the `::response` signal and take appropriate action.

**Parameters** **response\_id** (*str*) – response ID

**Raises** **TypeError** – when `response_id` is not a `str` type

#### **add\_button** (*button\_text=None, response\_id=None*)

Adds a button with the given `text` and sets things up so that clicking the button will emit the “response” signal with the given `response_id`.

The button is appended to the end of the dialog’s action area.

The button widget is returned, but usually you don’t need it.

##### **Parameters**

- **button\_text** (*str*) – text of button
- **response\_id** (*str*) – response ID for the button

**Returns** the `GLXCurses.Button` widget that was added.

**Return type** `GLXCurses.Button`

**Raises**

- **TypeError** – when `button_text` is not a str type
- **TypeError** – when `response_id` is not a int type

**add\_buttons** (\*args)

Adds more buttons, same as calling `Dialog.add_button()` repeatedly.

The data in arguments (args) must form a couple `button_text`, `response_id`.

Example: `Dialog.add_buttons('Hello.42', 42, 'Hello.43', 43, 'Hello.44', 44)`

Each button must have both text and response ID.

**Parameters** `args` – couple `button_text`, `response_id`

**add\_action\_widget** (*child=None, response\_id=None*)

Adds an activatable widget to the action area of a `GLXCurses.Dialog`, connecting a signal handler that will emit the “response” signal on the dialog when the widget is activated.

The widget is appended to the end of the dialog’s action area.

If you want to add a non-activatable widget, simply pack it into the `action_area` field of the `GLXCurses.Dialog` struct.

**Parameters**

- **child** (*GLXCurses.Widget*) – an activatable widget
- **response\_id** (*str*) – response ID for child

**Raises**

- **TypeError** – when `child` is not a `GLXCurses.Widget` instance
- **TypeError** – when `response_id` is not a str type

**set\_default\_response** (*response\_id=None*)

Sets the last widget in the dialog’s action area with the given `response_id` as the default widget for the dialog.

Pressing “Enter” normally activates the default widget.

**Parameters** `response_id` (*str*) – a response ID

**Raises** **TypeError** – when `response_id` is not a str type

**set\_response\_sensitive** (*response\_id=None, setting=None*)

Calls `gtk_widget_set_sensitive` (`widget`, `@setting`) for each widget in the dialog’s action area with the given `response_id`.

A convenient way to sensitize/desensitize dialog buttons.

**Parameters**

- **response\_id** (*str*) – a response ID
- **setting** (*bool*) – True for sensitive

**Raises**

- **TypeError** – when `response_id` is not a str type
- **TypeError** – when `setting` is not a bool type

**get\_response\_for\_widget** (*widget=None*)

Gets the response id of a `GLXCurses.Widget` in the action area of a `GLXCurses.Dialog`.

Note: That the return None if the widget is not found in action area.



**Parameters** `widget` (*GLXCurses.Widget*) – a widget in the action area of dialog

**Returns** the response id of *GLXCurses.Widget* , or *GLXC.RESPONSE\_NONE* if doesnt have a response id.

**Return type** int or *GLXC.RESPONSE\_NONE* or *None*

**get\_widget\_for\_response** (*response\_id=None*)

Gets the widget button that uses the given response ID in the action area of a dialog.

**Parameters** `response_id` (*str*) – the response ID used by the dialog widget

**Returns** the widget button that uses the given *response\_id* , or *None*.

**get\_action\_area** ()

has been deprecated since version GTK3.12, *GLXCurses* return the internal *\_action\_area*.

Here the structure:

```
““ [
    { 'widget': button_widget, 'response_id': response_id, 'default_response': False
    }, {
        'widget': button_widget, 'response_id': response_id, 'default_response': False
    }
]
```

Returns the action area of dialog .

**Returns** the action area.

**Return type** list

**get\_content\_area** ()

Returns the content area of dialog .

**Returns** the content area *GLXCurses* Box.

**Return type** *GLXCurses.VBox*

**close** ()

Signal emitted when the user uses a keybinding to close the dialog.

**update\_preferred\_sizes** ()

**draw\_widget\_in\_area** ()

Be here for be overwrite by every widget

**class** *GLXCurses.Application*

Bases: *glxeloop.bus.Bus*, *GLXCurses.Aera.Area*, *GLXCurses.libs.Spot.Spot*, *GLXCurses.libs.ApplicationHandlers.Handlers*

**Description**

Create a Application singleton instance.

That class have the role of a Controller and a NCurses Wrapper.

It have particularity to not be a *GLXCurses.Widget*, then have a tonne of function for be a fake *GLXCurses.Widget*.

From *GLXCurses* point of view everything start with it component. All widget will be display and store inside it component.

**get\_widget\_by\_id** (*widget\_id=None*)

**active\_window**

Gets the “active\_window” for the application.

The active *Window* is the one that was most recently focused (within the application).

This window may not have the focus at the moment if another application has it — this is just the most recently-focused window within this application.

**Returns** the active *Window*, or *None* if there isn’t one.

**Return type** *ChildElement* or *None*

**children**

Store the `children` property value

It property is use for store a stack of windows object use during choice of the active window

Default value: []

**Returns** `children` property value

**Return type** list

**app\_menu****menubar**

The MenuModel for the menubar.

**Returns** menubar property value

**Return type** GLXCurses.MenuBar or *None*

**register\_session****screensaver\_active****style**

The style of the Application, which contains information about how it will look (colors, etc).

The Application Style is impose to each widget

**Returns** a GLXCurses.Style instance

**Return type** GLXCurses.Style

**instance** = <GLXCurses.Application.Application object>

**statusbar****messagebar**

Sets the messagebar of application .

This can only be done in the primary instance of the application, after it has been registered. “startup” is a good place to call this.

**Returns** the messagebar property value

**Return type** GLXCurses.MessageBar or *None*

**toolbar****add\_window** (*window*)

Add a *Window* widget to the *Application* windows children’s list.

This call can only happen after the application has started; typically, you should add new application windows in response to the emission of the “activate” signal.

This call is equivalent to setting the “application” property of window to application .

Normally, the connection between the application and the window will remain until the window is destroyed, but you can explicitly remove it with `application.remove_window()`.

Galaxie-Curses will keep the application running as long as it has any windows.

**Parameters** `window` (*GLXCurses.Window*) – a window to add

**Raises** **TypeError** – if `window` parameter is not a *Window* type

**remove\_window** (*window*)

Remove a *Window* widget from the *Application* windows children's list.

Set "application" and "parent" attribute of the *GLXCurses.Window* to *None*.

**Parameters** `window` (*GLXCurses.Window*) – a window to add

**Raises** **TypeError** – if `window` parameter is not a *Window* type

**get\_window\_by\_id** (*identifier=None*)

Returns the *GtkApplicationWindow* with the given ID.

**Parameters** `identifier` (*int*) – an identifier number

**Returns** the window with ID `identifier`, or *None* if there is no window with this ID.

**Return type** *int* or *None*

**Raises** **TypeError** – when `identifier` is not a *int* type

**refresh** ()

Refresh the *NCurses* Screen, and redraw each contain widget's

It's a central refresh point for the entire application.

**check\_sizes** ()

Just a internal method for compute every size.

It consist to a serial of testable function call

**get\_mouse** ()

**eveloop\_input\_event** ()

**eveloop\_cmd** ()

**eveloop\_finalization** ()

**eveloop\_dispatch\_application** (*detailed\_signal, args*)

Flush Mainloop event to Child's father's for a *Widget*'s recursive event dispatch

**Parameters**

- **detailed\_signal** (*str*) – a string containing the signal name
- **args** (*list*) – additional parameters `arg1`, `arg2`

**eveloop\_keyboard\_interruption** ()

**class** *GLXCurses.Frame*

Bases: *GLXCurses.Bin.Bin*

**Description**

The frame widget is a bin that surrounds its child with a decorative frame and an optional label. If present, the label is drawn in a gap in the top side of the frame.

The position of the label can be controlled with *Frame.set\_label\_align()*.

**label**

Text of the frame's label.

Default value: None

**Returns** the `label` property value

**Return type** `str` or *None*

**label\_widget**

A widget to display in place of the usual frame label.

**Returns** A widget

**Return type** `GLXCurses.Label` or *None*

**label\_xalign**

The horizontal alignment of the label.

**Returns** The horizontal alignment of the label.

**Return type** `float`

**label\_yalign**

The vertical alignment of the label.

**Returns** The vertical alignment of the label.

**Return type** `float`

**shadow\_type**

Appearance of the frame border.

**Returns** The shadow type use by the frame

**Return type** `GLXCurses.GLXC.ShadowType`

**new** (*label=None*)

Create a new *Frame*, with optional label text .

If label is None, the label is omitted.

**Parameters** **label** (*str* or *None*) – the text to use as the label of the frame.

**Returns** a new *Frame* widget

**Return type** *Widget*

**set\_label** (*label*)

Sets the text of the label.

If label is None, the current label is removed.

**Parameters** **label** (*str* or *None*) – the text to use as the label of the frame.

**set\_label\_widget** (*label\_widget*)

Sets the label widget for the frame. This is the widget that will appear embedded in the top edge of the frame as a title.

**Parameters** **label\_widget** (*Widget*) – the new label widget

**set\_label\_align** (*xalign, yalign*)

Sets the alignment of the frame widget's label. The default values for a newly created frame are 0.0 and 0.5.

**Parameters**

- **xalign** (*float*) – The position of the label along the top edge of the widget. A value of 0.0 represents left alignment; 1.0 represents right alignment.
- **yalign** (*float*) – The y alignment of the label. A value of 0.0 aligns under the frame; 1.0 aligns above the frame. If the values are exactly 0.0 or 1.0 the gap in the frame won't be painted because the label will be completely above or below the frame.

**set\_shadow\_type** (*shadow\_type=None*)

Sets the shadow type for frame .

**Parameters** **shadow\_type** – the new :py:\_\_area\_data:ShadowType

**get\_label** ()

If the frame's label widget is a *Label*, returns the text in the label widget. (The frame will have a *Label* for the label widget if a non-NULL argument was passed when create the *Frame*.)

**Returns** the text in the label, or :py:\_\_area\_data:None if there was no label widget or the label widget was not a *Label* . This string is owned by GLXCurses and must not be modified or freed.

**Return type** str or *None*

**get\_label\_align** ()

Retrieves the X and Y alignment of the frame's label.

**See also:**

*Frame.set\_label\_align()*

**xalign:** X location of frame label

**yalign:** Y location of frame label

**Returns** xalign, yalign

**Return type** float, float

**get\_label\_widget** ()

Retrieves the label widget for the frame.

**See also:**

*Frame.set\_label\_widget()*

**Returns** the label widget, or NULL if there is none.

**Return type** *Widget* or :py:\_\_area\_data:None

**get\_shadow\_type** ()

Retrieves the shadow type of the frame.

**See also:**

*Frame.set\_shadow\_type()*

**Returns** the current shadow type of the frame.

**Return type** ShadowType

**draw\_widget\_in\_area** ()

Be here for be overwrite by every widget

**update\_preferred\_sizes** ()

```
class GLXCurses.MenuShell
    Bases: GLXCurses.Container.Container

    take_focus

    append (child=None)
        Adds a new GLXCurses.MenuItem to the end of the menu shell's item list.

        Returns A new GLXCurses.MenuItem to add

class GLXCurses.MenuBar
    Bases: GLXCurses.Box.Box, GLXCurses.libs.Dividable.Dividable, GLXCurses.MenuShell.MenuShell

    color

    info_label

    selected_menu

    selected_menu_item

    draw_widget_in_area ()
        White the menubar to the stdscr, the location is imposed to top left corner

    update_preferred_sizes ()

class GLXCurses.Menu
    Bases: GLXCurses.Window.Window, GLXCurses.libs.Movable.Movable, GLXCurses.Box.Box, GLXCurses.MenuShell.MenuShell

    color

    draw_widget_in_area ()
        Be here for be overwrite by every widget

    static remove_accel_group ()
        Not implemented

        Raises NotImplementedError – because AccelGroup is not implemented

    static add_accel_group ()
        Not implemented

        Raises NotImplementedError – because AccelGroup is not implemented

class GLXCurses.MenuItem
    Bases: GLXCurses.Widget.Widget

    accel_path
        Sets the accelerator path of the menu item, through which runtime changes of the menu item's accelerator
        caused by the user can be identified and saved to persistant storage.

        Default value: NULL

        Returns The accelerator path of the menu item

        Return type str

    label
        The text for the child label.

        Default value: ""

        Returns child label

        Return type str
```

**Raises `TypeError`** – When `label` property value is not str type or None

**`right_justified`**

Sets whether the menu item appears justified at the right side of a menu bar.

Default value: `False`

**Returns** `True` if the widget appears justified at the right side of a menu bar

**Return type** `bool`

**`text_short_cut`**

**`spacing`**

**`resized_text`**

**`resized_text_short_cut`**

**`is_accel`**

**`accelerator_size`**

**`color`**

**`draw()`**

**class** `GLXCurses.StatusBar`

Bases: `GLXCurses.Widget.Widget`

A `StatusBar` is usually placed along the bottom of an Application. It may provide a regular commentary of the application's status (as is usually the case in a web browser, for example), or may be used to simply output a message when the status changes, (when an upload is complete in an FTP client, for example).

Status bars in `GLXCurses` maintain a stack of messages. The message at the top of the each bar's stack is the one that will currently be displayed.

Any messages added to a `StatusBar`'s stack must specify a context id that is used to uniquely identify the source of a message. This context id can be generated by `GLXCurses.StatusBar.get_context_id()`, given a message and the `StatusBar` that it will be added to. Note that messages are stored in a stack, and when choosing which message to display, the stack structure is adhered to, regardless of the context identifier of a message.

One could say that a `StatusBar` maintains one stack of messages for display purposes, but allows multiple message producers to maintain sub-stacks of the messages they produced (via context ids).

Status bars are created using `GLXCurses.StatusBar.new()`.

Messages are added to the bar's stack with `GLXCurses.StatusBar.push()`.

The message at the top of the stack can be removed using `GLXCurses.StatusBar.pop()`.

A message can be removed from anywhere in the stack if its message id was recorded at the time it was added. This is done using `GLXCurses.StatusBar.remove()`.

**`new()`**

Creates a new `GLXCurses.StatusBar` ready for messages.

**Returns** the new `StatusBar`

**Return type** `GLXCurses.StatusBar`

**`get_context_id(context_description='Default')`**

Returns a new context identifier, given a description of the actual context.

**Parameters** `context_description` (`str`) – textual description of what context the new message is being used in. Default if none

**Returns** an `context_id` generate by `Utils.new_id()`

**Return type** `str`

**Raises** **TypeError** – When `context_description` is not a `str`

**push** (*context\_id*, *text*)

Push a new message onto the `StatusBar`'s stack.

**Parameters**

- **context\_id** (*str*) – a context identifier, as returned by `StatusBar.get_context_id()`
- **text** (*str*) – the message to add to the `StatusBar`

**Returns** a message identifier that can be used with `StatusBar.remove()`.

**Return type** `str`

**pop** (*context\_id*)

Removes the first message in the `StatusBar`'s stack with the given context id.

Note that this may not change the displayed message, if the message at the top of the stack has a different context id.

**Parameters** **context\_id** (*str*) – a context identifier, as returned by `StatusBar.get_context_id()`

**remove** (*context\_id*, *message\_id*)

Forces the removal of a message from a `StatusBar`'s stack. The exact **context\_id** and **message\_id** must be specified.

**Parameters**

- **context\_id** (*str*) – a context identifier, as returned by `StatusBar.get_context_id()`
- **message\_id** (*str*) – a message identifier, as returned by `StatusBar.push()`

**remove\_all** (*context\_id*)

Forces the removal of all messages from a `StatusBar`'s stack with the exact `context_id`.

**Parameters** **context\_id** (*str*) – a context identifier, as returned by `StatusBar.get_context_id()`

**draw** ()

Place the status bar from the end of the `stdscr` by look if it have a toolbar and a statusbar before

**update\_preferred\_sizes** ()

**class** `GLXCurses.MessageBar`

Bases: `GLXCurses.Widget.Widget`

A `MessageBar` is usually placed along the bottom of an Application. It may provide a regular commentary of the application's status (as is usually the case in a web browser, for example), or may be used to simply output a message when the status changes, (when an upload is complete in an FTP client, for example).

Status bars in `GLXCurses` maintain a stack of messages. The message at the top of the each bar's stack is the one that will currently be displayed.

Any messages added to a `StatusBar`'s stack must specify a context id that is used to uniquely identify the source of a message. This context id can be generated by `GLXCurses.StatusBar.get_context_id()`, given a message and the `StatusBar` that it will be added to. Note that messages are stored in a stack, and when choosing which message to display, the stack structure is adhered to, regardless of the context identifier of a message.

One could say that a `StatusBar` maintains one stack of messages for display purposes, but allows multiple message producers to maintain sub-stacks of the messages they produced (via context ids).



Status bars are created using `GLXCurses.MessageBar.new()`.

Messages are added to the bar's stack with `GLXCurses.MessageBar.push()`.

The message at the top of the stack can be removed using `GLXCurses.MessageBar.pop()`.

A message can be removed from anywhere in the stack if its message id was recorded at the time it was added. This is done using `GLXCurses.MessageBar.remove()`.

**new()**

Creates a new `GLXCurses.MessageBar` ready for messages.

**Returns** the new MessageBar

**Return type** `GLXCurses.MessageBar`

**get\_context\_id** (*context\_description*='Default')

Returns a new context identifier, given a description of the actual context.

**Parameters** **context\_description** (*str*) – textual description of what context the new message is being used in

**Returns** an context\_id generate by `Utils.new_id()`

**Return type** `str`

**Raises** **TypeError** – When context\_description is not a str

**push** (*context\_id*, *text*)

Push a new message onto the MessageBar's stack.

**Parameters**

- **context\_id** (*str*) – a context identifier, as returned by `MessageBar.get_context_id()`
- **text** (*str*) – the message to add to the MessageBar

**Returns** a message identifier that can be used with `MessageBar.remove()`.

**Return type** `str`

**pop** (*context\_id*)

Removes the first message in the MessageBar's stack with the given context id.

Note that this may not change the displayed message, if the message at the top of the stack has a different context id.

**Parameters** **context\_id** (*str*) – a context identifier, as returned by `MessageBar.get_context_id()`

**remove** (*context\_id*, *message\_id*)

Forces the removal of a message from a MessageBar's stack. The exact **context\_id** and **message\_id** must be specified.

**Parameters**

- **context\_id** (*str*) – a context identifier, as returned by `MessageBar.get_context_id()`
- **message\_id** (*str*) – a message identifier, as returned by `MessageBar.push()`

**remove\_all** (*context\_id*)

Forces the removal of all messages from a MessageBar's stack with the exact context\_id.

**Parameters** **context\_id** (*str*) – a context identifier, as returned by `MessageBar.get_context_id()`

**draw()**

Place the status bar from the end of the stdscr by look if it have a tool bar before

**update\_preferred\_sizes()**

**class** GLXCurses.**ToolBar**

Bases: *GLXCurses.Widget.Widget*

**draw()**

Draw the ToolBar widget

**labels**

Get the labels list, it contain items with dictionary with key 'id', 'text', 'end\_coord'

**Returns** The labels list

**Return type** list

**init\_button\_positions()**

Calculate positions of buttons; width is never less than 7

Else distribute the extra width in a way that the middle vertical line (between F5 and F6) aligns with the center of the screen.

The extra width is distributed in this order: F10, F5, F9, F4, ..., F6, F1.

**get\_button\_width** (*i=None*)

return width of one button

**Parameters** **i** (*int*) – button number it start to 0

**Returns** width of one button

**Return type** int

**Raises** **TypeError** – When **i** is not a int type

**get\_button\_by\_x\_coord** (*x=None*)

Return the button number by it X coordinate

**Parameters** **x** (*int*) – X coordinate value

**Returns** the button number

**Return type** int

**Raises** **TypeError** – When **x** is not a int type

**set\_label\_text** (*idx=None, text=None*)

Set the text to a button

**Parameters**

- **idx** (*int*) – The button id it start by 0
- **text** (*str*) – The text to set to it button

**update\_preferred\_sizes()**

**class** GLXCurses.**Misc**

Bases: *GLXCurses.Widget.Widget*

The Misc widget is an abstract widget which is not useful itself, but is used to derive subclasses which have alignment and padding attributes.

The horizontal and vertical padding attributes allows extra space to be added around the widget.

The horizontal and vertical alignment attributes enable the widget to be positioned within its allocated area. Note that if the widget is added to a container in such a way that it expands automatically to fill its allocated area, the alignment settings will not alter the widget's position.

Note that the desired effect can in most cases be achieved by using the “halign”, “valign” and “margin” properties on the child widget

**Warning:** To reflect this fact, all Misc API will be deprecated soon.

**xalign** - The horizontal alignment, from 0.0 to 1.0

**yalign** - The vertical alignment, from 0.0 to 1.0

**xpad** - The amount of space to add on the left and right of the widget, in characters

**ypad** - The amount of space to add above and below the widget, in characters

**xalign**

The horizontal alignment. A value of 0.0 means left alignment (or right on RTL locales); a value of 1.0 means right alignment (or left on RTL locales).

Allowed values: [0,1] Default value: 0.5

**Returns** The horizontal alignment value.

**Return type** float

**yalign**

The horizontal alignment. A value of 0.0 means left alignment (or right on RTL locales); a value of 1.0 means right alignment (or left on RTL locales).

Allowed values: [0,1] Default value: 0.5

**Returns** The horizontal alignment

**Return type** float

**xpad**

The amount of space to add on the left and right of the widget, in chars.

**Returns** The amount of space in chars

**Return type** int

**ypad**

**Returns** The amount of space in chars

**Raise** int

**class** GLXCurses.**Label**

Bases: *GLXCurses.Misc.Misc*, *GLXCurses.libs.Movable.Movable*

**angle**

The angle that the baseline of the label makes with the horizontal, in degrees, measured counterclockwise. An angle of 90 reads from from bottom to top, an angle of 270, from top to bottom.

Ignored if the label is selectable.

Allowed values: [0,360]

**Returns** angle that the baseline of the label

**Return type** int

**attributes**

A list of style attributes to apply to the text of the label.

**Returns** A list of style attributes

**Return type** list

**cursor\_position**

The current position of the insertion cursor in chars.

**Returns** The `cursor_position` property value

**Return type** int

**label**

The contents of the label.

If the string contains TXT Markdown, you will have to set the `use_markdown` property to True in order for the label to display the Markdown attributes.

See also `set_markdown()` for a convenience function that sets both this property and the `use_markdown` property at the same time.

If the string contains underlines acting as mnemonics, you will have to set the `use_underline` property to True in order for the label to display them.

**Returns** The content of the label

**Return type** str

**lines**

The number of lines to which an ellipsized, wrapping label should be limited. This property has no effect if the label is not wrapping or ellipsized.

Set this property to -1 if you don't want to limit the number of lines.

**Returns** The number of lines to which an ellipsized

**Return type** int

**max\_width\_chars**

The desired maximum width of the label, in characters.

If this property is set to -1, the width will be calculated automatically.

See the section on text layout for details of how `width_chars` and `max_width_chars` determine the width of ellipsized and wrapped labels.

**Returns** Maximum width of the label, in characters

**Return type** int

**mnemonic\_keyval**

The mnemonic accelerator key for this label.

Default value: 16777215

**mnemonic\_widget**

The GLXCurses.Widget to be activated when the label's mnemonic key is pressed.

**Returns** The GLXCurses.Widget to be activated or None if not set

:rtype GLXCurses.Widget or None

**pattern**

A string with `_` characters in positions correspond to characters in the text to underline.

**Returns** characters in the text use for underline

**Return type** str

**selectable**

Whether the GLXCurses.Label text can be selected with the mouse.

**Returns** True if GLXCurses.Label text can be selected

**Return type** bool

**selection\_bound**

The position of the opposite end of the selection from the cursor in chars.

**Returns** The position in chars

**Return type** int

**single\_line\_mode**

Whether the label is in single line mode. In single line mode, the height of the label does not depend on the actual text, it is always set to ascent + descent of the font.

This can be an advantage in situations where resizing the label because of text changes would be distracting, e.g. in a GLXCurses.StatusBar or GLXCurses.MessageBar.

Default value: False

**Returns** True if label is in single line mode

**Return type** bool

**track\_visited\_links**

Set this property to True to make the label track which links have been visited.

It will then apply the GLXC.STATE\_FLAG\_VISITED when rendering this link, in addition to GLXC.STATE\_FLAG\_LINK.

Default value: True

**Returns** True if label track which links have been visited

**Return type** bool

**use\_markdown**

The text of the label includes TXT Markdown.

Default value: False

**Returns** True if Markdown is used

**Return type** bool

**use\_underline**

If set, an underline in the text indicates the next character should be used for the mnemonic accelerator key.

Default value: False

**Returns** True if underline is display on text when use a mnemonic accelerator key

**Return type** bool

**width\_chars**

The desired width of the label, in characters. If this property is set to -1, the width will be calculated automatically.

See the section on text layout for details of how `width_chars` and `max_width_chars` determine the width of ellipsized and wrapped labels.

**Returns** The desired width of the label, in characters.

**Return type** `int`

#### **wrap**

If set, wrap lines if the text becomes too wide.

**Returns** True if wrap is in use

**Return type** `bool`

#### **wrap\_mode**

If line wrapping is on (see the `wrap` property) this controls how the line wrapping is done.

The default is `GLXC.WRAP_WORD`, which means wrap on word boundaries.

**Returns** How the line wrapping is done

**Return type** `GLXCurses.GLXC.WrapMode`

#### **new** (*string=None*)

Creates a new label with the given text inside it.

You can pass `None` to get an empty `GLXCurses.Label`.

**Parameters** **string** (*str or None*) – The text of the `GLXCurses.Label`.

**Returns** The new `GLXCurses.Label` it self

**Return type** `GLXCurses.Label`

#### **set\_text** (*string=None*)

Sets the text within the `GtkLabel` widget. It overwrites any text that was there before.

This function will clear any previously set mnemonic accelerators, and set the `use_underline` property to `False` as a side effect.

This function will set the `use_markdown` property to `False` as a side effect.

See also: `GLXCurses.Label().set_markdown()`

**Parameters** **string** (*str or None*) – The text you want to set

#### **set\_attributes** (*attributes=None*)

Sets a `GLXC.StateFlags`; the attributes in the list are applied to the label text.

The attributes set with this function will be applied and merged with any other attributes previously effected by way of the `use_underline` or `use_markup` properties.

While it is not recommended to mix markdown strings with manually set attributes, if you must; know that the attributes will be applied to the label after the markdown string is parsed.

**Parameters** **attributes** (*list or None*) – a `GLXC.StateFlags`

#### **set\_markdown** (*string=None*)

Parses `string` which is marked down with the text markdown language, setting the label's text and attribute list based on the parse results.

This function will set the `use_markup` property to `True` as a side effect.

If you set the label contents using the `label` property you should also ensure that you set the `use_markup` property accordingly.

See also: `GLXCurses.Label().set_text()`

**Parameters** `string` (*str*) – A markdown string (see text markdown format)

**set\_markdown\_with\_mnemonic** (*string*)

Parses *string* which is marked down with the text markdown language, setting the label's text and attribute list based on the parse results.

If characters in *string* are preceded by an underscore, they are underlined indicating that they represent a keyboard accelerator called a mnemonic.

The mnemonic key can be used to activate another GLXCurses.Widget, chosen automatically, or explicitly using GLXCurses.Label().set\_mnemonic\_widget().

**Parameters** `string` (*str*) – A markdown string (see text markdown format)

**set\_pattern** (*pattern=None*)

The pattern of underlines you want under the existing text within the GLXCurses.Label widget.

For example if the current text of the label says "FooBarBaz" passing a pattern of " \_ \_ \_ " will underline "Foo" and "Baz" but not "Bar".

**Parameters** `pattern` (*str or None*) – The pattern as described above.

**set\_justify** (*jtype=None*)

Sets the alignment of the lines in the text of the label relative to each other.

GLXCurses.GLXC.JUSTIFY\_LEFT is the default value when the widget is first created with GLXCurses.Label().new()

If you instead want to set the alignment of the label as a whole, use GLXCurses.Widget().set\_halign() instead.

GLXCurses.Label().set\_justify() has no effect on labels containing only a single line.

**Parameters** `jtype` (*str or None*) – a GLXCurses.GLXC.Justification

**set\_xalign** (*xalign=None*)

Sets the *xalign* property for label .

**Parameters** `xalign` (*float or None*) – the new *xalign* value, between 0 and 1

**set\_yalign** (*yalign=None*)

Sets the *yalign* property for label .

**Parameters** `yalign` (*float or None*) – the new *yalign* value, between 0 and 1

**set\_width\_chars** (*n\_chars=None*)

Sets the desired width in characters of label to *n\_chars* .

**Parameters** `n_chars` (*int or None*) – the new desired width, in characters.

**set\_max\_width\_chars** (*n\_chars*)

Sets the desired maximum width in characters of label to *n\_chars* .

**Parameters** `n_chars` (*int or None*) – the new desired maximum width, in characters.

**set\_line\_wrap** (*wrap=None*)

Toggles line wrapping within the GtkLabel widget. `True` makes it break lines if text exceeds the widget's size. `False` lets the text get cut off by the edge of the widget if it exceeds the widget size.

Note that setting line wrapping to `TRUE` does not make the label wrap at its parent container's width, because GLXCurses widgets conceptually can't make their requisition depend on the parent container's size. For a label that wraps at a specific position, set the label's width using GLXCurses.Widget().set\_size\_request()

**Parameters** `wrap` (*bool or None*) – `True` if wrap is enable

**set\_line\_wrap\_mode** (*wrap\_mode=None*)

If line wrapping is on (see GLXCurses.Label().set\_line\_wrap()) this controls how the line wrapping is done. The default is GLXCurses.GLXC.WRAP\_WORD which means wrap on word boundaries.

**Parameters** **wrap\_mode** (*str* or *None*) – the line wrapping mode

**set\_lines** (*lines=None*)

Sets the number of lines to which an ellipsized, wrapping label should be limited. This has no effect if the label is not wrapping or ellipsized.

Set this to -1 if you don't want to limit the number of lines.

**Parameters** **lines** (*int* or *None*) – the desired number of lines, or -1

**get\_mnemonic\_keyval** ()

If the label has been set so that it has an mnemonic key this function returns the keyval used for the mnemonic accelerator.

If there is no mnemonic set up it returns *None*.

**Returns** ord() keyval usable for accelerators, or *None*

**Return type** *int* or *None*

**get\_selectable** ()

Gets the value set by GLXCurses.Label().set\_selectable().

**Returns** *True* if the user can copy text from the label

**get\_text** ()

Fetches the text from a label widget, as displayed on the screen.

This does not include any embedded underlines indicating mnemonics or markdown.

(See GLXCurses.Label().get\_label())

**Returns** the text in the label widget. This is the internal string used by the label, and must not be modified.

**Return type** *str* or *None*

**new\_with\_mnemonic** (*string=None*)

Creates a new GtkLabel, containing the text in *str* .

If characters in *str* are preceded by an underscore, they are underlined. If you need a literal underscore character in a label, use ‘\_\_’ (two underscores). The first underlined character represents a keyboard accelerator called a mnemonic. The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `gtk_label_set_mnemonic_widget()`.

If GLXCurses.Label().set\_mnemonic\_widget() is not called, then the first activatable ancestor of the GLXCurses.Label will be chosen as the mnemonic widget. For instance, if the label is inside a button or menu item, the button or menu item will automatically become the mnemonic widget and be activated by the mnemonic.

**Parameters** **string** (*str* or *None*) – The text of the label, with an underscore in front of the mnemonic character.

**select\_region** (*start\_offset=None, end\_offset=None*)

Selects a range of characters in the label, if the label is selectable. See GLXCurses.Label().set\_selectable().

If the label is not selectable, this function has no effect. If *start\_offset* or *end\_offset* are -1, then the end of the label will be substituted.

**Parameters**



- **start\_offset** (*int*) – start offset (in characters not bytes)
- **end\_offset** (*int*) – end offset (in characters not bytes)

**Raises** `TypeError` – when

**set\_use\_underline** (*setting*)

**get\_use\_underline** ()

**set\_mnemonic\_widget** (*widget*)

**get\_mnemonic\_widget** ()

**set\_selectable** (*setting=None*)

Selectable labels allow the user to select text from the label, for copy-and-paste.

**Parameters** **setting** (*bool* or *None*) – True to allow selecting text in the label

**set\_text\_with\_mnemonic** (*string*)

Sets the label's text from the string *str* . If characters in *str* are preceded by an underscore, they are underlined indicating that they represent a keyboard accelerator called a mnemonic.

The mnemonic key can be used to activate another widget, chosen automatically, or explicitly using `GLXCurses.Label().set_mnemonic_widget()`.

**Parameters** **string** (*str*) – a string

**draw\_widget\_in\_area** ()

Be here for be overwrite by every widget

**update\_preferred\_sizes** ()

**get\_justify** ()

Returns the justification of the label.

**See also:**

`Label.set_justify()` for set the justification.

**Returns** the justification

**Return type** `GLXCurses.GLXC.Justification`

**get\_line\_wrap** ()

The `get_line_wrap()` method returns the value of the “wrap” property.

If “wrap” is True the lines in the label are automatically wrapped. See `set_line_wrap()`.

**Returns** True if wrap is enable

**Return type** `bool`

**get\_width\_chars** ()

The `get_width_chars()` method returns the value of the `width-chars` property that specifies the desired width of the label in characters.

**Returns** width of the label in characters

**Return type** `int`

**set\_single\_line\_mode** (*single\_line\_mode*)

**get\_single\_line\_mode** ()

**get\_max\_width\_chars** ()

```
get_line_wrap_mode()
```

```
class GLXCurses.ProgressBar
```

```
    Bases: GLXCurses.Widget.Widget, GLXCurses.libs.Movable.Movable
```

```
    color (pos=0)
```

```
    text
```

```
    value
```

```
    show_text
```

```
    draw_widget_in_area()
```

```
        Be here for be overwrite by every widget
```

```
    update_preferred_sizes()
```

```
class GLXCurses.HSeparator
```

```
    Bases: GLXCurses.Widget.Widget, GLXCurses.libs.Movable.Movable
```

The `GLXCurses.HSeparator` widget is a horizontal separator, used to visibly separate the widgets within a window.

It displays a horizontal line.

```
    draw_widget_in_area()
```

```
        Be here for be overwrite by every widget
```

```
    update_preferred_sizes()
```

```
class GLXCurses.VSeparator
```

```
    Bases: GLXCurses.Widget.Widget, GLXCurses.libs.Movable.Movable
```

The `GLXCurses.VSeparator` widget is a vertical separator, used to visibly separate the widgets within a window.

It displays a vertical line.

```
    draw_widget_in_area()
```

```
        Be here for be overwrite by every widget
```

```
    update_preferred_sizes()
```

```
class GLXCurses.EntryBuffer
```

```
    Bases: GLXCurses.Object.Object
```

`EntryBuffer` — Text buffer for *GLXCurses.Entry*

#### Description

The *GLXCurses.EntryBuffer* class contains the actual text displayed in a *GLXCurses.Entry* widget.

A single *GLXCurses.EntryBuffer* object can be shared by multiple *GLXCurses.Entry* widgets which will then share the same text content, but not the cursor position, visibility attributes, etc.

*GLXCurses.EntryBuffer* may be derived from. Such a derived class might allow text to be stored in an alternate location, such as non-pageable memory, useful in the case of important passwords. Or a derived class could integrate with an application's concept of undo/redo.

```
    max_length
```

```
    length
```

```
        The length property
```

```
        Allowed values: <= 65535
```

```
        Default value: 0
```

**Returns** The length (in characters) of the text in buffer.

**Return type** int

**text**

The text property

**Returns** The contents of the buffer.

**Return type** char

**new** (*initial\_chars=None, n\_initial\_chars=-1*)

Create a new *GLXCurses.EntryBuffer* object.

Optionally, specify initial text to set in the buffer.

**Parameters**

- **initial\_chars** – initial buffer text, or None
- **n\_initial\_chars** – number of characters in initial\_chars , or -1

**Returns** the new EntryBuffer

**Return type** *GLXCurses.EntryBuffer.EntryBuffer*

**Raises**

- **TypeError** – if initial\_chars is not printable string or None
- **TypeError** – if n\_initial\_chars is not int or -1

**get\_text** ()

Retrieves the contents of the buffer.

The memory pointer returned by this call will not change unless this object emits a signal, or is finalized.

**Returns** a pointer to the contents of the widget as a string. This string points to internally allocated storage in the buffer and must not be freed, modified or stored.

**Return type** str

**set\_text** (*chars=”, n\_chars=-1*)

Sets the text in the buffer.

This is roughly equivalent to calling EntryBuffer.delete\_text() and EntryBuffer.insert\_text().

---

**Note:** n\_chars is in characters, not in bytes.

---

**Parameters**

- **chars** (*str*) – the new text
- **n\_chars** (*int*) – the number of characters in text , or -1

**Raises**

- **TypeError** – if chars is not str
- **TypeError** – if n\_chars is not int or -1

**get\_bytes** ()

Retrieves the length in bytes of the buffer.

**See also:**

EntryBuffer.get\_length().

**Returns** The byte length of the buffer.

**Return type** int

**get\_length()**

Retrieves the length in characters of the buffer.

**Returns** The number of characters in the buffer.

**Return type** int

**get\_max\_length()**

Retrieves the maximum allowed length of the text in buffer .

**See also:**

EntryBuffer.set\_max\_length().

**Returns** the maximum allowed number of characters in EntryBuffer, or 0 if there is no maximum.

**Return type** int

**set\_max\_length(max\_length=0)**

Sets the maximum allowed length of the contents of the buffer. If the current contents are longer than the given length, then they will be truncated to fit.

**Parameters** **max\_length** (*int*) – The maximum length of the entry buffer, or 0 for no maximum. (other than the maximum length of entries.) The value passed in will be clamped to the range 0-65536.

**Raises** **TypeError** – if `max_length` is not int

**insert\_text(position=0, chars="", n\_chars=-1)**

Inserts `n_chars` characters of `chars` into the contents of the buffer, at position `position` .

If `n_chars` is negative, then characters from `chars` will be inserted until a null-terminator is found. If `position` or `n_chars` are out of bounds, or the maximum buffer text length is exceeded, then they are coerced to sane values.

---

**Note:** The position and length are in characters, not in bytes.

---

#### Parameters

- **position** (*int*) – The position at which to insert text.
- **chars** (*str*) – The text to insert into the buffer.
- **n\_chars** (*int*) – The length of the text in characters, or -1

**Returns** The number of characters actually inserted.

**Return type** int

**Raises**

- **TypeError** – if `position` is not int
- **TypeError** – if `chars` is not printable str

- **TypeError** – if `n_chars` is not int

**delete\_text** (*position=None, n\_chars=-1*)

Deletes a sequence of characters from the buffer. `n_chars` characters are deleted starting at `position`. If `n_chars` is negative, then all characters until the end of the text are deleted.

If `position` or `n_chars` are out of bounds, then they are coerced to sane values.

---

**Note:** The positions are specified in characters, not bytes..

---

#### Parameters

- **position** (*int*) – Position at which to delete text
- **n\_chars** (*int*) – Number of characters to delete

**Returns** The number of characters deleted.

**Return type** int

#### Raises

- **TypeError** – if `position` is not int
- **TypeError** – if `n_chars` is not int

**class** GLXCurses.**Editable**

Bases: object

**select\_region** (*editable=None, start\_pos=None, end\_pos=None*)

Selects a region of text. The characters that are selected are those characters at positions from `start_pos` up to, but not including `end_pos`. If `end_pos` is negative, then the characters selected are those characters from `start_pos` to the end of the text.

Note that positions are specified in characters, not bytes.

#### Parameters

- **editable** (*GLXCurses.Editable or None*) – a GLXCurses.Editable
- **start\_pos** (*int or None*) – start of region
- **end\_pos** (*int or None*) – end of region

#### Raises

- **TypeError** – if `start_pos` is not a int type or None.
- **TypeError** – if `end_pos` is not a int type or None.
- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.

**get\_selection\_bounds** (*editable=None*)

Retrieves the selection bound of the editable. `start_pos` will be filled with the start of the selection and `end_pos` with end. If no text was selected both will be identical and FALSE will be returned.

Note that positions are specified in characters, not bytes.

**Parameters** **editable** (*GLXC.Editable or None*) – a GLXC.Editable

**Returns** True if an area is selected, False otherwise

**Return type** bool

**Raises**

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.

**insert\_text** (*editable=None, new\_text=None, new\_text\_length=-1, position=None*)

Inserts `new_text_length` bytes of `new_text` into the contents of the widget, at position `position` .

Note that the position is in characters, not in bytes.

The function updates position to point after the newly inserted text.

**Parameters**

- **editable** (*GLXC.Editable or None*) – a GLXC.Editable
- **new\_text** (*str*) – the text to append
- **new\_text\_length** (*int*) – the length of the text in bytes, or -1
- **position** (*int or None*) – location of the position text will be inserted at. None for insert at actual position.

**Raises**

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.
- **TypeError** – if `new_text` is not a str or None.
- **TypeError** – if `new_text_length` is not a int or None.
- **TypeError** – if `position` is not a int or None.

**delete\_text** (*editable=None, start\_pos=None, end\_pos=None*)

Deletes a sequence of characters. The characters that are deleted are those characters at positions from `start_pos` up to, but not including `end_pos` .

If `end_pos` is negative, then the characters deleted are those from `start_pos` to the end of the text.

**Parameters**

- **editable** (*GLXC.Editable or None*) – a GLXC.Editable
- **start\_pos** (*int or None*) – start position
- **end\_pos** (*int or None*) – end position

**Raises**

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.
- **TypeError** – if `start_pos` is not a int type or None.
- **TypeError** – if `end_pos` is not a int type or None.

**get\_chars** (*editable=None, start\_pos=None, end\_pos=None*)

Retrieves a sequence of characters. The characters that are retrieved are those characters at positions from `start_pos` up to, but not including `end_pos` .

If `end_pos` is negative, then the characters retrieved are those characters from `start_pos` to the end of the text.

Note that positions are specified in characters, not bytes.

#### Parameters

- **editable** (*GLXC.Editable* or *None*) – a *GLXC.Editable*
- **start\_pos** (*int*) – start of text
- **end\_pos** (*int*) – end of text

**Returns** a pointer to the contents of the widget as a string. This string is allocated by the *GLXC.Editable* implementation and should be freed by the caller.

#### Raises

- **TypeError** – if *editable* is not a valid *GLXCurses* type.
- **ImportError** – if *editable* is not a instance of *GLXCurses.Editable*.
- **TypeError** – if *start\_pos* is not a *int* type or *None*.
- **TypeError** – if *end\_pos* is not a *int* type or *None*.

**cut\_clipboard** (*editable=None*)

Removes the contents of the currently selected content in the editable and puts it on the clipboard.

**Parameters** *editable* (*GLXCurses.Editable* or *None*) – a instance of *GLXCurses.Editable*.

#### Raises

- **TypeError** – if *editable* is not a valid *GLXCurses* type.
- **TypeError** – if *editable* is not a instance of *GLXCurses.Editable*.

**copy\_clipboard** (*editable=None*)

Copies the contents of the currently selected content in the editable and puts it on the clipboard.

**Parameters** *editable* (*GLXCurses.Editable* or *None*) – a *GLXCurses.Editable*

#### Raises

- **TypeError** – if *editable* is not a valid *GLXCurses* type.
- **TypeError** – if *editable* is not a instance of *Editable*.

**paste\_clipboard** (*editable=None*)

Pastes the content of the clipboard to the current position of the cursor in the editable.

**Parameters** *editable* (*GLXC.Editable* or *None*) – a *GLXC.Editable*

#### Raises

- **TypeError** – if *editable* is not a valid *GLXCurses* type.
- **TypeError** – if *editable* is not a instance of *GLXCurses.Editable*.

**delete\_selection** (*editable=None*)

Deletes the currently selected text of the editable. This call doesnt do anything if there is no selected text.

**Parameters** *editable* (*GLXC.Editable*) – a Class Name contain on the list *GLXC.Editable*

#### Raises

- **TypeError** – if *editable* is not a valid *GLXCurses* type.
- **TypeError** – if *editable* is not a instance of *GLXCurses.Editable*.

**set\_position** (*editable=None, position=-1*)

Sets the cursor position in the editable to the given value.

The cursor is displayed before the character with the given (base 0) index in the contents of the editable. The value must be less than or equal to the number of characters in the editable.

A value of -1 indicates that the position should be set after the last character of the editable.

Note that position is in characters, not in bytes.

**Parameters**

- **editable** (*GLXC.Editable*) – a Class Name contain on the list GLXC.Editable
- **position** (*int*) – the position of the cursor

**Raises**

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.
- **TypeError** – if `position` is not a int type.

**get\_position** (*editable=None*)

Retrieves the current position of the cursor relative to the start of the content of the editable.

Note that this position is in characters, not in bytes.

**Parameters** **editable** (*GLXC.Editable*) – a Class Name contain on the list GLXC.Editable

**Returns** the cursor position

**Return type** int

**Raises**

- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.

**set\_editable** (*editable=None, is\_editable=True*)

Determines if the user can edit the text in the editable widget or not.

**Parameters**

- **editable** (*GLXC.Editable*) – a Class Name contain on the list GLXC.Editable
- **is\_editable** (*bool*) – True if the user is allowed to edit the text in the widget

**Raises**

- **TypeError** – if `is_editable` is not a int type.
- **TypeError** – if `editable` is not a valid GLXCurses type.
- **TypeError** – if `editable` is not a instance of GLXCurses.Editable.

**get\_editable** (*editable=None*)

Retrieves whether editable is editable.

See GLXCurses.Editable.set\_editable().

**Parameters** **editable** (*GLXC.Editable*) – a Class Name contain on the list GLXC.Editable

**Returns** True if editable is editable.



**Raises**

- **TypeError** – if `editable` is not a valid `GLXCurses` type.
- **TypeError** – if `editable` is not an instance of `GLXCurses.Editable`.

**class** `GLXCurses.Entry`

Bases: `GLXCurses.Widget.Widget`, `GLXCurses.Editable.Editable`, `GLXCurses.libs.Movable.Movable`

Entry — A single line text entry field

**Description**

The `GLXCurses.Entry` widget is a single line text entry widget. A fairly large set of key bindings are supported by default. If the entered text is longer than the allocation of the widget, the widget will scroll so that the cursor position is visible.

When using an entry for passwords and other sensitive information, it can be put into “password mode” using `GLXCurses.Entry.set_visibility()`. In this mode, entered text is displayed using a “invisible” character. By default, `GLXCurses` picks the best invisible character that is available in the current font, but it can be changed with `GLXCurses.Entry.set_invisible_char()`. `GLXCurses` displays a warning when Caps Lock or input methods might interfere with entering text in a password entry. The warning can be turned off with the “caps-lock-warning” property.

**activates\_default**

Whether to activate the default widget (such as the default button in a dialog) when Enter is pressed.

Default Value: `False`

**Returns** `True` if the widget is the default widget

**Return type** `bool`

**Raises** **TypeError** – When `activates_default` property value is not a `bool` type or `None`

**attributes**

A list of attributes to apply to the text of the entry.

This is mainly useful to change the size or weight of the text.

The Attribute’s `start_index` and `end_index` must refer to the `EntryBuffer` text, i.e. without the preedit string.

Default Value: `[]`

**Returns** attributes to apply to the text of the entry

**Return type** `list`

**Raises** **TypeError** – When `attributes` property value is not a `list` type or `None`

**buffer**

Text buffer object which actually stores entry text.

Default Value: `GLXCurses.EntryBuffer()`

**Returns** The Text buffer object

**Return type** `GLXCurses.EntryBuffer`

**Raises** **TypeError** – When `buffer` property value is not a `GLXCurses.EntryBuffer` instance or `None`

**caps\_locks\_warning**

Whether password entries will show a warning when Caps Lock is on.

Note that the warning is shown using a secondary icon, and thus does not work if you are using the secondary icon position for some other purpose.

Default Value: `True`

**Returns** `True` if a warning icon is display

**Return type** `bool`

**Raises `TypeError`** – When `caps_locks_warning` property value is not a `bool` type or `None`

#### **completion**

The auxiliary completion object to use with the entry.

**Returns** The Completion object

**Return type** `GLXCurses.EntryCompletion`

**Raises `TypeError`** – When `completion` property value is not a `GLXCurses.EntryCompletion` instance or `None`

#### **cursor\_position**

The current position of the insertion cursor in chars.

- Allowed values: `[0,65535]`
- Default Value: `0`

**Returns** `True` if the widget is the default widget

**Return type** `bool`

**Raises `TypeError`** – When `cursor_position` property value is not a `int` type or `None`

#### **editable**

Whether the entry contents can be edited.

**Returns** `True` if the entry contents can be edited

**Return type** `bool`

**Raises `TypeError`** – When `editable` property value is not a `bool` type or `None`

#### **enable\_emoji\_completion**

Whether to suggest Emoji replacements.

Default value: `False`

**Returns** `True` if the completion of emoji is enable

**Return type** `bool`

**Raises `TypeError`** – When `completion` property value is not a `bool` type or `None`

#### **has\_frame**

`FALSE` removes outside bevel from entry.

Default value: `True`

**Returns** `False` removes outside bevel from entry.

**Return type** `bool`

**Raises `TypeError`** – When `has_frame` property value is not a `bool` type or `None`

**im\_module**

Which IM (input method) module should be used for this entry. See GLXCurses.IMContext.

Setting this to a non-NULL value overrides the system-wide IM module setting. See the GLXCurses.Settings “glxcurses\_im\_module” property.

Default value: None

**Returns** IM (input method) as text

**Return type** str or *None*

**Raises** **TypeError** – When im\_module property value is not a str type or None

**inner\_border**

Sets the text area’s border between the text and the frame.

**Returns** a GLXCurses.GLXC.BorderStyle item

**Return type** str

**Raises**

- **TypeError** – When inner\_border property value is not a str type or None
- **Value** – When inner\_border property value is not a GLXCurses.GLXC.BorderStyle

**input\_hints**

Additional hints (beyond “input\_purpose”) that allow input methods to fine-tune their behaviour.

**Returns** GLXC.InputHints item

**Return type** str

**Raises**

- **TypeError** – When input\_hints property value is not a str type or None
- **Value** – When input\_hints property value is not a GLXCurses.GLXC.InputHints

**input\_purpose**

The purpose of this text field.

This property can be used by on-screen keyboards and other input methods to adjust their behaviour.

Note that setting the purpose to GLXC.INPUT\_PURPOSE\_PASSWORD or GLXC.INPUT\_PURPOSE\_PIN is independent from setting “visibility”.

**Returns** a GLXCurses.GLXC.InputPurpose item

**Return type** str

**Raises**

- **TypeError** – When input\_purpose property value is not a str type or None
- **Value** – When input\_purpose property value is not a GLXCurses.GLXC.InputPurpose item

**invisible\_char**

The invisible character is used when masking entry contents (in “password mode”). When it is not explicitly set with the “invisible-char” property, GLXCurses determines the character to use from a list of possible candidates, depending on availability in the current font.

This style property allows the theme to prepend a character to the list of candidates.

Default value: ‘\*’

**Returns** the invisible character

**Return type** str

**Raises** **TypeError** – When `invisible_char` property value is not a str type or None

**invisible\_char\_set**

Whether the invisible char has been set for the GLXCurses.Entry.

Default value: False

**Returns** True if char has been set for the GLXCurses.Entry.

**Return type** bool

**Raises** **TypeError** – When `invisible_char_set` property value is not a bool type or None

**max\_length**

Maximum number of characters for this entry. Zero if no maximum.

Default value: 0

**Returns** Maximum number of characters for this entry

**Return type** int

**Raises** **TypeError** – When `max_length` property value is not a bool type or None

**max\_width\_chars**

The desired maximum width of the entry, in characters. If this property is set to -1, the width will be calculated automatically.

Allowed values:  $\geq -1$

Default value: -1

**Returns** The desired maximum width of the entry

**Return type** int

**Raises** **TypeError** – When `max_width_chars` property value is not a int type or None

**overwrite\_mode**

If text is overwritten when typing in the GLXCurses.Entry.

Default value: False

**Returns** True If text is overwritten when typing

**Return type** bool

**Raises** **TypeError** – When `overwrite_mode` property value is not a bool type or None

**placeholder\_text**

The text that will be displayed in the GtkEntry when it is empty and unfocused.

Default value: None

**Returns** IM (input method) as text

**Return type** str or *None*

**Raises** **TypeError** – When `im_module` property value is not a str type or None

**populate\_all**

If `populate_all` is True, the **populate-popup** signal is also emitted for touch popups.

Default value: False

**Returns** True If **populate-popup** signal is also emitted for touch popups.

**Return type** bool

**Raises** **TypeError** – When `populate_all` property value is not a bool type or None

#### **primary\_icon\_activatable**

Whether the primary icon is activatable.

GLXCurses emits the “icon-press” and “icon-release” signals only on sensitive, activatable icons.

Sensitive, but non-activatable icons can be used for purely informational purposes.

Default value: True

**Returns** True If emit “icon-press” and “icon-release” on sensitive, activable icon.

**Return type** bool

**Raises** **TypeError** – When `primary_icon_activatable` property value is not a bool type or None

#### **primary\_icon\_gicon**

The GIcon to use for the primary icon for the entry.

Default value: None

**Returns** something like a icon name

**Return type** str

**Raises** **TypeError** – When `primary_icon_gicon` property value is not a str type or None

#### **primary\_icon\_name**

The icon name to use for the primary icon for the entry.

Default value: None

**Returns** something like a icon name

**Return type** str

**Raises** **TypeError** – When `primary_icon_gicon` property value is not a str type or None

#### **primary\_icon\_sensitive**

Whether the primary icon is sensitive.

An insensitive icon appears grayed out. GLXCurses does not emit the “icon-press” and “icon-release” signals and does not allow DND from insensitive icons.

An icon should be set insensitive if the action that would trigger when clicked is currently not available.

Default value: True

**Returns** If True an insensitive icon appears grayed out.

**Return type** bool

**Raises** **TypeError** – When `primary_icon_sensitive` property value is not a bool type or None

#### **primary\_icon\_tooltip\_markdown**

The contents of the tooltip on the primary icon, which is marked up with the text markdown language

**Returns** The contents of the tooltip with markdown format

**Return type** str or *None*

**primary\_icon\_tooltip\_text**

The contents of the tooltip on the primary icon, which is marked up with the text format

**Returns** The contents of the tooltip with text format

**Return type** str or *None*

**progress\_fraction**

The current fraction of the task that's been completed.

Allowed values: [0,1]

Default value: 0

**Returns** The the fraction value

**Return type** float

**draw\_widget\_in\_area()**

Be here for be overwrite by every widget

**new()**

Creates a new entry.

**Returns** A new GLXCurses Entry Widget

**Return type** GLXCurses.Widget

**new\_with\_buffer(buffer=None)**

Creates a new entry with the specified text buffer.

---

**Note:** `Utils.is_valid_id()` and `Utils.new_id()` are used for identify if the `buffer` is a Galaxie-Curses component. That GLXCurses ID is automatically generate at the widget creation.

---

**Parameters** **buffer** – The buffer to use for the new `GLXCurses.Entry`.

**Returns** A Entry Buffer object.

**Return type** `GLXCurses.Entry`

**Raises**

- **TypeError** – if `buffer` is not `GLXCurses.EntryBuffer` Type
- **TypeError** – if `buffer` haven't a valid GLXCurses ID

**get\_buffer()**

Get the GLXCurses.EntryBuffer object which holds the text for this widget.

**Returns** A EntryBuffer object.

**Return type** GLXCurses.Widget

**set\_buffer(buffer=None)**

Set the EntryBuffer object which holds the text for this widget.

**Parameters** **buffer** – The buffer to use for the GLXCurses.Entry.

**set\_text(text)**

Sets the text in the widget to the given value, replacing the current contents.

**See also:**

`GLXCurses.EntryBuffer().set_text()`

**Parameters** **text** (*String*) – The new text

**get\_text** ()

Retrieves the contents of the entry widget. See also GLXCurses.Editable.get\_chars().

This is equivalent to: “ self.buffer = GLXCurses.EntryBuffer() self.buffer.get\_text() “ :return: A pointer to the contents of the widget as a string. This string points to internally allocated storage in the widget and must not be freed, modified or stored. :rtype: String

**get\_text\_length** ()

Retrieves the current length of the text in entry .

This is equivalent to: “ self.buffer = GLXCurses.EntryBuffer() self.buffer.get\_length() “

**Returns** The current number of characters in GtkEntry, or 0 if there are none.

**Return type** Int in range 0-65536

**set\_visibility** (*visible=None*)

Sets whether the contents of the entry are visible or not. When visibility is set to FALSE, characters are displayed as the invisible char, and will also appear that way when the text in the entry widget is copied elsewhere.

By default, GLXCurse picks the best invisible character available in the current font, but it can be changed with set\_invisible\_char().

---

**Note:** You probably want to set “input\_purpose” to glx.INPUT\_PURPOSE\_PASSWORD or glx.INPUT\_PURPOSE\_PIN to inform input methods about the purpose of this entry, in addition to setting visibility to FALSE.

---

**Parameters** **visible** (*bool*) – True if the contents of the entry are displayed as plaintext

**Raises** **TypeError** – if *visible* is not boolean type

**set\_invisible\_char** (*ch='\*'*)

Sets the character to use in place of the actual text when set\_visibility() has been called to set text visibility to FALSE.

---

**Note:** this is the character used in “password mode” to show the user how many characters have been typed.

---

By default, GLXCurse picks the best invisible char available in the current font.

---

**Note:** If you set the invisible char to 0, then the user will get no feedback at all; there will be no text on the stdscr as they type

---

**Parameters** **ch** (*str*) – a character

**Raises** **TypeError** – if *ch* is not printable str

**unset\_invisible\_char** ()

” Unset the invisible char previously set with set\_invisible\_char(). So that the default invisible char is used again.

**set\_max\_length** (*max=None*)

Sets the maximum allowed length of the contents of the widget. If the current contents are longer than the given length, then they will be truncated to fit.

**This is equivalent to:** `self.buffer = GLXCurses.EntryBuffer() self.buffer.set_max_length()`

**Parameters** **max** (*int*) – The maximum length of the entry, or 0 for no maximum. (other than the maximum length of entries.) The value passed in will be clamped to the range 0-65536.

**get\_activates\_default** ()

Retrieves the value set by `set_activates_default()`.

**Returns** TRUE if the entry will activate the default widget

**Return type** bool

**get\_has\_frame** ()

Gets the value set by `set_has_frame()`.

**Returns** whether the entry has a beveled frame

**Return type** bool

**get\_inner\_border** ()

This function returns the entry's "inner-border" property. See `set_inner_border()` for more information.

GLXC.BorderStyle Members:

GLXC.BORDER\_STYLE\_NONE No visible border GLXC.BORDER\_STYLE\_SOLID A single line segment GLXC.BORDER\_STYLE\_INSET Looks as if the content is sunken into the canvas GLXC.BORDER\_STYLE\_OUTSET Looks as if the content is coming out of the canvas GLXC.BORDER\_STYLE\_HIDDEN Same as `glxc.BORDER_STYLE_NONE` GLXC.BORDER\_STYLE\_DOTTED A series of round dots GLXC.BORDER\_STYLE\_DASHED A series of square-ended dashes GLXC.BORDER\_STYLE\_DOUBLE Two parallel lines with some space between them GLXC.BORDER\_STYLE\_GROOVE Looks as if it were carved in the canvas GLXC.BORDER\_STYLE\_RIDGE Looks as if it were coming out of the canvas

**Returns** a GLXC.BorderStyle type Constant or GLXC.BORDER\_STYLE\_NONE if none was set

**Return type** str

**get\_width\_chars** ()

Gets the value set by `set_width_chars()`

**Returns** number of chars to request space for, or negative if unset

**get\_max\_width\_chars** ()

Retrieves the desired maximum width of entry , in characters.

`set_max_width_chars()`.

**Returns** the maximum width of the entry, in characters

**Return type** int

**set\_activates\_default** (*setting*)

If setting is True, pressing Enter in the entry will activate the default widget for the window containing the entry.

This usually means that the dialog box containing the entry will be closed, since the default widget is usually one of the dialog buttons.



(For experts: if setting is True, the entry calls `activate_default()` on the window containing the entry, in the default handler for the “activate” signal.)

**Parameters** `setting` (*bool*) – True to activate window’s default widget on Enter keypress

**Raises** `TypeError` – if `setting` is not bool type

**set\_has\_frame** (*setting=True*)

Sets whether the entry has a beveled frame around it.

**Parameters** `setting` (*bool*) – False removes outside bevel from entry

**Raises** `TypeError` – if `setting` is not bool type

**set\_inner\_border** (*border='BORDER\_STYLE\_NONE'*)

Sets entry’s inner-border property to `border`, or clears it if `None` is passed. The inner-border is the area around the entry’s text, but inside its frame.

If set, this property overrides the inner-border style property. Overriding the style-provided border is useful when you want to do in-place editing of some text in a canvas or list widget, where pixel-exact positioning of the entry is important.

### **GLXC.BorderStyle**

Describes how the border of a UI element should be rendered.

**Members:** `GLXC.BORDER_STYLE_NONE` No visible border `GLXC.BORDER_STYLE_SOLID` A single line segment `GLXC.BORDER_STYLE_INSET` Looks as if the content is sunken into the canvas `GLXC.BORDER_STYLE_OUTSET` Looks as if the content is coming out of the canvas `GLXC.BORDER_STYLE_HIDDEN` Same as `glxc.BORDER_STYLE_NONE` `GLXC.BORDER_STYLE_DOTTED` A series of round dots `GLXC.BORDER_STYLE_DASHED` A series of square-ended dashes `GLXC.BORDER_STYLE_DOUBLE` Two parallel lines with some space between them `GLXC.BORDER_STYLE_GROOVE` Looks as if it were carved in the canvas `GLXC.BORDER_STYLE_RIDGE` Looks as if it were coming out of the canvas

**Parameters** `border` (*str*) – a valid `GLXC.BorderStyle`

**Raises**

- **TypeError** – if `border` is not str type
- **TypeError** – if `border` is not a valid `GLXC.BorderStyle`

**set\_width\_chars** (*n\_chars=-1*)

Changes the size request of the entry to be about the right size for `n_chars` characters. Note that it changes the size request, the size can still be affected by how you pack the widget into containers.

If `n_chars` is -1, the size reverts to the default entry size.

**Parameters** `n_chars` (*int*) – width in chars

**Raises** `TypeError` – if `n_chars` is not int type

**set\_max\_width\_chars** (*n\_chars=-1*)

Sets the desired maximum width in characters of entry

**Parameters** `n_chars` (*int*) – the new desired maximum width, in characters

**Raises** `TypeError` – if `n_chars` is not int type

**get\_invisible\_char** ()

Retrieves the character displayed in place of the real characters for entries with visibility set to false.

**See also:**

`set_invisible_char()`.

**Returns** the current invisible char, or 0, if the entry does not show invisible text at all.

**set\_alignment** (*xalign=0.0*)

Sets the alignment for the contents of the entry. This controls the horizontal positioning of the contents when the displayed text is shorter than the width of the entry.

**Parameters** **xalign** (*float*) – The horizontal alignment, from 0 (left) to 1 (right). Reversed for RTL layouts

**Raises** **TypeError** – if `xalign` is not float type

**get\_alignment** ()

Gets the value set by `GLXCurses.Entry.set_alignment()`.

**Returns** The horizontal alignment, from 0 (left) to 1 (right).

**Return type** float

**set\_placeholder\_text** (*text=None*)

Sets text to be displayed in entry when it is empty and unfocused. This can be used to give a visual hint of the expected contents of the `GLXCurses.Entry`.

---

**Note:** that since the placeholder text gets removed when the entry received focus, using this feature is a bit problematic if the entry is given the initial focus in a window. Sometimes this can be worked around by delaying the initial focus setting until the first key event arrives.

---

**Parameters** **text** (*str or None*) – a string to be displayed when entry is empty and unfocused, or None.

**Raises** **TypeError** – if `text` is not str or None type

**get\_placeholder\_text** ()

Retrieves the text that will be displayed when entry is empty and unfocused

**Returns** a pointer to the placeholder text as a string. This string points to internally allocated storage in the widget and must not be freed, modified or stored.

**set\_overwrite\_mode** (*overwrite=False*)

Sets whether the text is overwritten when typing in the `GLXCurses.Entry`.

**Parameters** **overwrite** (*bool*) – new value

**Raises** **TypeError** – if `overwrite` is not bool type

**get\_overwrite\_mode** ()

Gets the value set by `GLXCurses.Entry.set_overwrite_mode()`.

**Returns** whether the text is overwritten when typing.

**Return type** bool

**get\_layout** ()

**Raises** **NotImplementedError** – GLXCurses don't get Pango management

**get\_layout\_offsets** ()

**Raises** **NotImplementedError** – GLXCurses don't get Pango management

**layout\_index\_to\_text\_index** ()

**Raises `NotImplementedError`** – GLXCurses don't get Pango management

**`text_index_to_layout_index()`**

**Raises `NotImplementedError`** – GLXCurses don't get Pango management

**`set_attributes()`**

**Raises `NotImplementedError`** – GLXCurses don't get Pango management

**`get_attributes()`**

**Raises `NotImplementedError`** – GLXCurses don't get Pango management

**`get_max_length()`**

Retrieves the maximum allowed length of the text in entry . See `GLXCurses.Entry.set_max_length()`.

This is equivalent to getting entry 's `GLXCurses.EntryBuffer` and calling `GLXCurses.EntryBuffer.get_max_length()` on it.

**Returns** the maximum allowed number of characters in `GLXCurses.Entry`, or 0 if there is no maximum.

**Return type** `int`

**`get_visibility()`**

Retrieves whether the text in entry is visible.

---

**Note:** `GLXCurses.EntryBuffer.set_visibility()`

---

**Returns** True if the text is currently visible

**Return type** `bool`

**`set_completion(completion=None)`**

Sets completion to be the auxiliary completion object to use with entry . All further configuration of the completion mechanism is done on completion using the `GtkEntryCompletion` API. Completion is disabled if completion is set to `None`.

**Parameters** `completion` (`GLXCurses.EntryCompletion.EntryCompletion` or `None`) – The `GLXCurses.EntryCompletion.EntryCompletion` or `None`.

**Raises `TypeError`** – when completion is not `GLXCurses.EntryCompletion` or `None`

**`get_completion()`**

Returns the auxiliary completion object currently in use by entry .

**Returns** The auxiliary completion object currently in use by entry .

**Return type** `GLXCurses.EntryCompletion` or `None`

**`set_cursor_hadjustment(adjustment=None)`**

Hooks up an adjustment to the cursor position in an entry, so that when the cursor is moved, the adjustment is scrolled to show that position. See `scrolled_window_get_hadjustment()` for a typical way of obtaining the adjustment.

The adjustment has to be in char units and in the same coordinate system as the entry.

**Parameters** `adjustment` (`GLXCurses.Adjustment.Adjustment` or `None`) – an adjustment which should be adjusted when the cursor is moved, or `None`.

**Raises `TypeError`** – when completion is not `GLXCurses.Adjustment.Adjustment` or `None`

**get\_cursor\_hadjustment()**

Retrieves the horizontal cursor adjustment for the entry. See GLXCurses.Adjustment.Adjustment.set\_cursor\_hadjustment().

**Returns** the horizontal cursor adjustment, or NULL if none has been set.

**Return type** *GLXCurses.Adjustment.Adjustment* or *None*

**set\_progress\_fraction(fraction=0.0)**

Causes the entry's progress indicator to "fill in" the given fraction of the bar. The fraction should be between 0.0 and 1.0, inclusive.

**Parameters** **fraction** (*float*) – fraction of the task that's been completed

**Raises** **TypeError** – when fraction is not float type

**get\_progress\_fraction()**

Returns the current fraction of the task that's been completed. See GLXCurses.Entry.Entry.set\_progress\_fraction().

**Returns** a fraction from 0.0 to 1.0

**Return type** *float*

**set\_progress\_pulse\_step(fraction=0.1)**

Sets the fraction of total entry width to move the progress bouncing block for each call to GLXCurses.Entry.Entry.progress\_pulse().

**Parameters** **fraction** (*float*) – fraction between 0.0 and 1.0

**Raises** **TypeError** – when fraction is not float type

**get\_progress\_pulse\_step()**

Retrieves the pulse step set with GLXCurses.Entry.Entry.set\_progress\_pulse\_step().

**Returns** a fraction from 0.0 to 1.0

**Return type** *float*

**progress\_pulse()**

Indicates that some progress is made, but you don't know how much. Causes the entry's progress indicator to enter "activity mode," where a block bounces back and forth. Each call to GLXCurses.Entry.Entry.progress\_pulse() causes the block to move by a little bit (the amount of movement per pulse is determined by GLXCurses.Entry.Entry.set\_progress\_pulse\_step()).

**Raises** **NotImplementedError** – GLXCurses don't deal with that yet.

**im\_context\_filter\_keypress()**

Allow the GLXCurses.Entry input method to internally handle key press and release events.

If this function returns TRUE, then no further processing should be done for this key event.

See GLXCurses.Entry.im\_context\_filter\_keypress().

Note that you are expected to call this function from your handler when overriding key event handling. This is needed in the case when you need to insert your own key handling between the input method and the default key event handling of the GLXCurses.Entry.

See GLXCurses.Entry.text\_view\_reset\_im\_context() for an example of use.

**Raises** **NotImplementedError** – GLXCurses don't deal with that yet.

**reset\_im\_context()**

Reset the input method context of the entry if needed.

This can be necessary in the case where modifying the buffer would confuse on-going input method behavior.

**Raises `NotImplementedError`** – GLXCurses don't deal with that yet.

**`get_tabs()`**

Gets the tabstops that were set on the entry using `gtk_entry_set_tabs()`, if any.

**Raises `NotImplementedError`** – GLXCurses don't deal with that yet.

**`set_tabs()`**

Sets a `PangoTabArray`; the tabstops in the array are applied to the entry text.

**Raises `NotImplementedError`** – GLXCurses don't deal with that yet.

**`set_icon_from_pixbuf()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`set_icon_from_stock()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`set_icon_from_icon_name()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`set_icon_from_gicon()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`get_icon_storage_type()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`get_icon_pixbuf()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`get_icon_stock()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`get_icon_name()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`get_icon_gicon()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`set_icon_activatable()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`get_icon_activatable()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`set_icon_sensitive()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`get_icon_sensitive()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**`get_icon_at_pos()`**

**Raises `NotImplementedError`** – GLXCurses don't get icon's management

**set\_icon\_tooltip\_text()**

Raises **NotImplementedError** – GLXCurses don't get icon's management

**get\_icon\_tooltip\_text()**

Raises **NotImplementedError** – GLXCurses don't get icon's management

**set\_icon\_tooltip\_markup()**

Raises **NotImplementedError** – GLXCurses don't get icon's management

**get\_icon\_tooltip\_markup()**

Raises **NotImplementedError** – GLXCurses don't get icon's management

**set\_icon\_drag\_source()**

Raises **NotImplementedError** – GLXCurses don't get icon's management

**get\_current\_icon\_drag\_source()**

Raises **NotImplementedError** – GLXCurses don't get icon's management

**get\_icon\_area()**

Raises **NotImplementedError** – GLXCurses don't get icon's management

**set\_input\_purpose()**

Raises **NotImplementedError** – GLXCurses don't get icon's management

**get\_input\_purpose()**

Raises **NotImplementedError** – GLXCurses don't get icon's management

**set\_input\_hints()**

Raises **NotImplementedError** – GLXCurses don't deal with hints

**get\_input\_hints()**

Raises **NotImplementedError** – GLXCurses don't deal with hints

**grab\_focus\_without\_selecting()**

Causes entry to have keyboard focus.

It behaves like `gtk_widget_grab_focus()`, except that it doesn't select the contents of the entry. You only want to call this on some special entries which the user usually doesn't want to replace all text in, such as search-as-you-type entries.

**get\_states()**

**update\_preferred\_sizes()**

**class GLXCurses.EntryCompletion**

Bases: *GLXCurses.Object.Object*

EntryCompletion — Completion functionality for *GLXCurses.Entry*

### Properties

**inline\_completion**

WDetermines whether the common prefix of the possible completions should be inserted automatically in the entry. Note that this requires `text-column` to be set, even if you are using a custom match function.

**Type** bool

**Flags** Read / Write

**Default value** False

#### **inline\_selection**

Determines whether the possible completions on the popup will appear in the entry as you navigate through them.

**Type** bool

**Flags** Read / Write

**Default value** False

#### **minimum\_key\_length**

Minimum length of the search key in order to look up matches.

**Type** bool

**Flags** Read / Write

**Allowed values**  $\geq 0$

**Default value** 1

#### **model**

The model to find matches in.

**Type** TreeModel

**Flags** Read / Write

#### **popup\_completion**

Determines whether the possible completions should be shown in a popup window.

**Type** bool

**Flags** Read / Write

**Default value** True

#### **popup\_single\_match**

Determines whether the completions popup window will shown for a single possible completion. You probably want to set this to False if you are using inline completion.

**Type** bool

**Flags** Read / Write

**Default value** True

#### **text\_column**

The column of the model containing the strings. Note that the strings must be UTF-8.

**Type** int

**Flags** Read / Write

**Allowed values**  $\geq -1$

**Default value** -1

#### **new()**

Creates a new EntryCompletion object.

**Returns** A new GLXCurse Entry Completion object

**Return type** GLXCurse.EntryCompletion

**class** GLXCurses.RangeBases: *GLXCurses.Widget.Widget*

Range — Base class for widgets which visualize an adjustment

**Properties****adjustment**

The GLXCurses.Adjustment.Adjustment that contains the current value of this range object.

**Type** GLXCurses.Adjustment.Adjustment**Flags** Read / Write / Construct**fill\_level**

The fill level (e.g. prebuffering of a network stream). See GLXCurses.Adjustment.Adjustment.set\_fill\_level().

**Type** float**Flags** Read / Write**Default value** 1.79769e+308**inverted**

Invert direction slider moves to increase range value.

**Type** bool**Flags** Read / Write**Default value** False**model**

The model to find matches in.

**Type** TreeModel**Flags** Read / Write**lower\_stepper\_sensitivity**

The sensitivity policy for the stepper that points to the adjustment's lower side.

**Type** bool**Flags** Read / Write**Default value** GLXCurses.GLXC.SENSITIVITY\_AUTO**restrict\_to\_fill\_level**

The restrict-to-fill-level property controls whether slider movement is restricted to an upper boundary set by the fill level. See GLXCurses.Adjustment.Adjustment.set\_restrict\_to\_fill\_level().

**Type** bool**Flags** Read / Write**Default value** True**round-digits**

The number of digits to round the value to when it changes, or -1. See "change-value".

**Type** int**Flags** Read / Write**Allowed values**  $\geq -1$



**Default value** -1

#### **show\_fill\_level**

The show-fill-level property controls whether fill level indicator graphics are displayed on the trough. See `GLXCurses.Adjustment.Adjustment.set_show_fill_level()`.

**Type** bool

**Flags** Read / Write

**Default value** False

#### **upper\_stepper\_sensitivity**

The sensitivity policy for the stepper that points to the adjustment's upper side.

**Type** `GLXCurses.GLXC.SensitivityType`

**Flags** Read / Write

**Default value** `GLXC.SENSITIVITY_AUTO`

#### **get\_fill\_level()**

Gets the current position of the fill level indicator.

**Returns** The current fill level

**Return type** int

#### **get\_restrict\_to\_fill\_level()**

Gets whether the range is restricted to the fill level.

**Returns** True if range is restricted to the fill level.

**Return type** bool

#### **get\_show\_fill\_level()**

Gets whether the range displays the fill level graphically.

**Returns** True if range shows the fill level.

**Return type** bool

#### **set\_fill\_level(fill\_level=1.79769e+308)**

Set the new position of the fill level indicator.

The “fill level” is probably best described by its most prominent use case, which is an indicator for the amount of pre-buffering in a streaming media player. In that use case, the value of the range would indicate the current play position, and the fill level would be the position up to which the file/stream has been downloaded.

This amount of prebuffering can be displayed on the range's trough and is themeable separately from the trough. To enable fill level display, use `GLXCurses.Range.Range.set_show_fill_level()`. The range defaults to not showing the fill level.

Additionally, it's possible to restrict the range's slider position to values which are smaller than the fill level. This is controller by `GLXCurses.Range.Range.set_restrict_to_fill_level()` and is by default enabled.

**Parameters** **fill\_level** (*float*) – the new position of the fill level indicator

**Raises** **TypeError** – if `fill_level` is not a float type

#### **set\_restrict\_to\_fill\_level(restrict\_to\_fill\_level=True)**

Sets whether the slider is restricted to the fill level. See `GLXCurses.Range.Range.set_fill_level()` for a general description of the fill level concept.

**Parameters** `restrict_to_fill_level` (*bool*) – Whether the fill level restricts slider movement.

**Raises** **TypeError** – if `restrict_to_fill_level` is not a bool type

**set\_show\_fill\_level** (*show\_fill\_level*)

Sets whether a graphical fill level is show on the trough. See `GLXCurses.Range.Range.set_fill_level()` for a general description of the fill level concept.

**Parameters** `show_fill_level` (*bool*) – Whether a fill level indicator graphics is shown.

**Raises** **TypeError** – if `show_fill_level` is not a bool type

**get\_adjustment** ()

Get the `GLXCurses.Adjustment.Adjustment` which is the “model” object for `GLXCurses.Range.Range`. See `GLXCurses.Range.Range.set_adjustment()` for details.

That because `GLXCurses.Range.Range` use internally a `GLXCurses.Adjustment.Adjustment`, the Attribute `adjustment` should never been touch or unreferenced.

**Returns** A `GLXCurses.Adjustment.Adjustment`

**Return type** *GLXCurses.Adjustment.Adjustment*

**set\_adjustment** (*adjustment=None*)

Sets the adjustment to be used as the “model” object for this range widget.

The adjustment indicates the current range value, the minimum and maximum range values, the step/page increments used for keybindings and scrolling, and the page size. The page size is normally 0 for `GtkScale` and nonzero for `Scrollbar`, and indicates the size of the visible area of the widget being scrolled. The page size affects the size of the scrollbar slider.

**Parameters** `adjustment` (*GLXCurses.Adjustment.Adjustment or None*) – `GLXCurses.Adjustment.Adjustment` or `None` for create a new one

**Raises** **TypeError** – if `adjustment` is not a `GLXCurses.Adjustment.Adjustment` or `None`

**get\_inverted** ()

Gets the value set by `GLXCurses.Range.Range.set_inverted()`.

**Returns** True if the range is inverted

**Return type** bool

**set\_inverted** (*setting=False*)

Ranges normally move from lower to higher values as the slider moves from top to bottom or left to right. Inverted ranges have higher values at the top or on the right rather than on the bottom or left.

**Parameters** `setting` (*bool*) – True to invert the range

**Raises** **TypeError** – if `setting` is not a bool type

**get\_value** ()

Gets the current value of the range.

**Returns** current value of the range.

**Return type** float

**set\_value** (*value=<class 'float'>*)

Sets the current value of the range; if the value is outside the minimum or maximum range values, it will be clamped to fit inside them. The range emits the “value-changed” signal if the value changes.

**Parameters** `value` (*float*) – new value of the range

**Raises** **TypeError** – if `value` is not a float type

**set\_increments** (*step=<class 'float'>, page=<class 'float'>*)

Sets the step increment and page increment for the range. The step increment is used when the user clicks the GLXCurses.Scrollbar.Scrollbar arrows or moves GLXCurses.Scale.Scale via arrow keys. The page size is used for example when moving via Page Up or Page Down keys.

Care: the GTK documentation is wrong compare to the the GTK Code source: <https://github.com/GNOME/gtk/blob/master/gtk/gtkrange.c#L1001>

That is step\_increment and page\_increment it be upgrade via a Adjustment.configure() and not step size and page size.

#### Parameters

- **step** (*float*) – the new step increment
- **page** (*float*) – the new page increment

**set\_range** (*min=None, max=None*)

Sets the allowable values in the GLXCurses.Range.Range, and clamps the range value to be between min and max . (If the range has a non-zero page size, it is clamped between min and max - page-size.)

#### Parameters

- **min** (*float*) – minimum range value
- **max** (*float*) – maximum range value

**get\_round\_digits** ()

Gets the number of digits to round the value to when it changes. See “change-value”.

**Returns** the number of digits to round to

**Return type** int

**set\_round\_digits** (*round\_digits=-1*)

Sets the number of digits to round the value to when it changes. See “change-value”.

**Parameters** **round\_digits** (*int*) – the precision in digits, or -1

**Raises** **TypeError** – if round\_digits is not a a int type

**set\_lower\_stepper\_sensitivity** (*sensitivity='AUTO'*)

Sets the sensitivity policy for the stepper that points to the ‘lower’ end of the GLXCurses.Range.Range’s adjustment.

Allowed Type:

**The arrow is made insensitive if the thumb is at the end** GLXCurses.GLXC.SENSITIVITY\_AUTO = ‘AUTO’

**The arrow is always sensitive** GLXCurses.GLXC.SENSITIVITY\_ON = ‘ON’

**The arrow is always insensitive** GLXCurses.GLXC.SENSITIVITY\_OFF = ‘OFF’

**Parameters** **sensitivity** (*GLXCurses.GLXC.SensitivityType*) – the lower stepper’s sensitivity policy.

**Raises** **TypeError** – if sensitivity is not a GLXCurses.GLXC.SensitivityType

**get\_lower\_stepper\_sensitivity** ()

Gets the sensitivity policy for the stepper that points to the ‘lower’ end of the GLXCurses.Range.Range’s adjustment.

**Returns** The lower stepper’s sensitivity policy.

**Return type** GLXCurses.GLXC.SensitivityType

**set\_upper\_stepper\_sensitivity** (*sensitivity='AUTO'*)

Sets the sensitivity policy for the stepper that points to the 'upper' end of the GLXCurses.Range.Range's adjustment.

**Parameters** **sensitivity** (*GLXCurses.GLXC.SensitivityType*) – The upper stepper's sensitivity policy.

**Raises** **TypeError** – if *sensitivity* is not a GLXCurses.GLXC.SensitivityType

**get\_upper\_stepper\_sensitivity** ()

Gets the sensitivity policy for the stepper that points to the 'upper' end of the GLXCurses.Range.Range's adjustment.

**Returns** The upper stepper's sensitivity policy.

**Return type** GLXCurses.GLXC.SensitivityType

**get\_flippable** ()

Gets the value set by GLXCurses.Range.Range.set\_flippable().

**Returns** True if the range is flippable

**Return type** bool

**set\_flippable** (*flippable=False*)

If a range is flippable, it will switch its direction if it is horizontal and its direction is GLXCurses.GLXC.TEXT\_DIR\_RTL.

**Parameters** **flippable** (*bool*) – True to make the range flippable

**Raises** **TypeError** – if *flippable* is not a bool type.

**get\_range\_rect** ()

This function returns the area that contains the range's trough and its steppers, in widget->window coordinates.

This function is useful mainly for Range subclasses.

**Returns** list(x, y, width, height)

**Return type** list

**get\_slider\_range** (*slider\_start=None, slider\_end=None*)

This function returns sliders range along the long dimension, in widget->window coordinates.

This function is useful mainly for Range subclasses.

If *slider\_start* or *slider\_end* are not None it will return the value.

Example:

*slider\_start=None, slider\_end=None* return list [None; None]

*slider\_start=1, slider\_end=1* return list [the\_calculated\_slider\_start; the\_calculated\_slider\_end]

**Parameters**

- **slider\_start** – return location for the slider's start, or None
- **slider\_end** – return location for the slider's end, or None

**get\_slider\_size\_fixed** ()

This function is useful mainly for GtkRange subclasses.

See GLXCurses.Range.Range.set\_slider\_size\_fixed().

**Returns** whether the range's slider has a fixed size.

**Return type** bool

**set\_slider\_size\_fixed** (*size\_fixed=<class 'bool'>*)

Sets whether the range's slider has a fixed size, or a size that depends on its adjustment's page size.

This function is useful mainly for GtkRange subclasses.

**Parameters** *size\_fixed* (bool) – True to make the slider size constant

**Raises** **TypeError** – if *size\_fixed* is not a bool type.

**class** GLXCurses.Actionable

Bases: object

Actionable — An interface for widgets that can be associated with actions

### Known Implementations

**Actionable is implemented by GLXC.Actionable and contain a list of widget** Button, CheckButton, CheckMenuItem, ColorButton, FontButton, ImageMenuItem, LinkButton, ListBoxRow, LockButton, MenuButton, MenuItem, MenuToolButton, ModelButton, RadioButton, RadioMenuItem, RadioToolButton, ScaleButton, SeparatorMenuItem, Switch, TearoffMenuItem, ToggleButton, ToggleToolButton, ToolButton, VolumeButton.

**action\_name**

**action\_target**

**get\_action\_name** ()

Gets the action name for actionable .

See set\_action\_name() for more information.

**Returns** the action name, or None if unset.

**Return type** str or *None*

**set\_action\_name** (*action\_name=None*)

Specifies the name of the action with which this widget should be associated. If *action\_name* is NULL then the widget will be unassociated from any previous action.

Usually this function is used when the widget is located (or will be located) within the hierarchy of a ApplicationWindow.

Names are of the form “win.save” or “app.quit” for actions on the containing ApplicationWindow or its associated GLXCurses.Application, respectively.

This is the same form used for actions in the GMenu associated with the window.

**Parameters** *action\_name* (str or *None*) – an action name, or None.

**Raises** **TypeError** – if *action\_name* is not a str type or None

**get\_action\_target\_value** ()

Gets the current target value of actionable .

See gtk\_actionable\_set\_action\_target\_value() for more information.

**Returns** the current target value.

**Return type** GLXCurses.Object or *None*

**set\_action\_target\_value** (*target\_value=None*)

Gets the current target value of actionable .

See `gtk_actionable_set_action_target_value()` for more information.

**Parameters** `target_value` (*GLXCurses.Object* or *None*) – the target value, or `NULL`

**class** `GLXCurses.FileSelect`

**Bases:** `GLXCurses.Widget.Widget`, `GLXCurses.libs.FileChooserFunctions.FileChooserUtils`

**item\_it\_can\_be\_display**

Get the number of item it can be display, as set by `_set_item_it_can_be_display()`.

**Returns** The number of item it can be display

**Return type** `int`

**item\_scroll\_pos**

Get the number of item it can be display, as set by `_set_item_it_can_be_display()`.

**Returns** The Position on the scroll list

**Return type** `int`

**selected\_item\_pos**

Position of the selected item.

**Returns** The Position on the scroll list

**Return type** `int`

**selected\_item\_info\_list**

Get the selected file information's list.

**The line\_info information's store position:** `item_name_text` in position [0] `item_path_sys` in position [1] `item_size_text` in position [2] `item_time_text` in position [3]

**Returns** information's about selected item.

**Return type** `dict`

`x_pos_history_next_label`

`x_pos_history_list_label`

`x_pos_history_prev_label`

`x_pos_history_actual_path`

`x_pos_history_actual_path_allowed_size`

`x_pos_title_mtime`

`x_pos_title_size`

`x_pos_title_name`

`x_pos_line_start`

`x_pos_line_stop`

`y_pos_history`

`y_pos_titles`

`y_pos_items`

`name_column_width`

```

    mtime_column_width
    size_column_width
    draw_widget_in_area()
        Be here for be overwrite by every widget
    update_preferred_sizes()
class GLXCurses.Image
    Bases: GLXCurses.Misc.Misc, GLXCurses.libs.File.File, GLXCurses.libs.Colors.Colors
    image_object
        Store the modified image
        Returns
    data
        Get data property
        Returns image data as a list
        Return type list
    hsp_debug
        Get hsp_debug property
        Returns image hsp_debug as a list
        Return type list
    width_max
        Get the width_max property value
        Returns width_max property value
        Return type int or None
    width_original
        Get the width_original property value
        Returns width_original property value
        Return type int or None
    height_max
        Get the height_max property value
        Returns height_max property value
        Return type int or None
    height_original
        Get the height_original property value
        it property is use when the widget discover image size
        Returns height_original property value
        Return type int or None
    is_resized
        Whether the image will be resized directly on the widget.
        Returns True or False
        Return type bool

```

```
load_image (path=None)

draw_widget_in_area ()
    Be here for be overwrite by every widget

to_data ()

class GLXCurses.ImageConvert
    Bases: GLXCurses.libs.File.File

    data
        Get data property

        Returns image data as a list

        Return type list

    hsp_debug
        Get hsp_debug property

        Returns image hsp_debug as a list

        Return type list

    width_max
        Get the width_max property value

        Returns width_max property value

        Return type int or None

    width_original
        Get the width_original property value

        Returns width_original property value

        Return type int or None

    height_max
        Get the height_max property value

        Returns height_max property value

        Return type int or None

    height_original
        Get the height_original property value

        it property is use when the widget discover image size

        Returns height_original property value

        Return type int or None

    is_resized
        Whether the image will be resized directly on the widget.

        Returns True or False

        Return type bool

    load_image (path=None)
```



## 6.3 Basic Types

Reference document: <https://docs.python.org/2/library/types.html>

### None

**NoneType** The type of `None`

### bool

**BooleanType** The type of the *bool* values `True` and `False`

### int

**IntType** The type of integers (e.g. `1`)

### long

**LongType** The type of long integers (e.g. `1L`)

### float

**FloatType** The type of floating point numbers (e.g. `1.0`)

### str

**StringType** The type of character strings (e.g. `'Spam'`)

### unicode

**UnicodeType** The type of Unicode character strings (e.g. `u'Spam'`)

### tuple

**TupleType** The type of tuples (e.g. `(1, 2, 3, 'Spam')`)

### list

**ListType** The type of lists (e.g. `[0, 1, 2, 3]`)

### dict

**DictType** The type of dictionaries (e.g. `{'Bacon': 1, 'Ham': 0}`)

## 6.4 Constant Value

### 6.4.1 enum BaselinePosition

#### BaselinePosition

Whenever a container has some form of natural row it may align children in that row along a common typographical baseline. If the amount of vertical space in the row is taller than the total requested height of the baseline-aligned children then it can use a `GtkBaselinePosition` to select where to put the baseline inside the extra available space.

**glxc.BASELINE\_POSITION\_TOP** Align the baseline at the top

**glxc.BASELINE\_POSITION\_CENTER** Center the baseline

**glxc.BASELINE\_POSITION\_BOTTOM** Align the baseline at the bottom

## 6.4.2 enum DeleteType

### DeleteType

See also: “delete-from-cursor”.

**glxc.DELETE\_CHARS** Delete characters.

**glxc.DELETE\_WORD\_ENDS** Delete only the portion of the word to the left/right of cursor if we’re in the middle of a word.

**glxc.DELETE\_WORDS** Delete words.

**glxc.DELETE\_DISPLAY\_LINES** Delete display-lines. Display-lines refers to the visible lines, with respect to the current line breaks. As opposed to paragraphs, which are defined by line breaks in the input.

**glxc.DELETE\_DISPLAY\_LINE\_ENDS** Delete only the portion of the display-line to the left/right of cursor.

**glxc.DELETE\_PARAGRAPH\_ENDS** Delete to the end of the paragraph. Like C-k in Emacs (or its reverse).

**glxc.DELETE\_PARAGRAPHS** Delete entire line. Like C-k in pico.

**glxc.DELETE\_WHITESPACE** Delete only whitespace. Like M-in Emacs.

## 6.4.3 enum DirectionType

### DirectionType

Focus movement types.

**glxc.DIR\_TAB\_FORWARD** Move forward.

**glxc.DIR\_TAB\_BACKWARD** Move backward.

**glxc.DIR\_UP** Move up.

**glxc.DIR\_DOWN** Move down.

**glxc.DIR\_LEFT** Move left.

**glxc.DIR\_RIGHT** Move right.

## 6.4.4 enum Justification

### Justification

Used for justifying the text inside a GtkLabel widget. (See also GtkAlignment).

**glxc.JUSTIFY\_LEFT** The text is placed at the left edge of the label.

**glxc.JUSTIFY\_RIGHT** The text is placed at the right edge of the label.

**glxc.JUSTIFY\_CENTER** The text is placed in the center of the label.

**glxc.JUSTIFY\_FILL** The text is placed is distributed across the label.

## 6.4.5 enum MovementStep

### **MovementStep**

Movements

**glxc.MOVEMENT\_LOGICAL\_POSITIONS** Move forward or back by graphemes  
**glxc.MOVEMENT\_VISUAL\_POSITIONS** Move left or right by graphemes  
**glxc.MOVEMENT\_WORDS** Move forward or back by words  
**glxc.MOVEMENT\_DISPLAY\_LINES** Move up or down lines (wrapped lines)  
**glxc.MOVEMENT\_DISPLAY\_LINE\_ENDS** Move to either end of a line  
**glxc.MOVEMENT\_PARAGRAPHS** Move up or down paragraphs (newline-ended lines)  
**glxc.MOVEMENT\_PARAGRAPH\_ENDS** Move to either end of a paragraph  
**glxc.MOVEMENT\_PAGES** Move by pages  
**glxc.MOVEMENT\_BUFFER\_ENDS** Move to ends of the buffer  
**glxc.MOVEMENT\_HORIZONTAL\_PAGES** Move horizontally by pages

## 6.4.6 enum Orientation

### **Orientation**

Represents the orientation of widgets and other objects which can be switched between horizontal and vertical orientation on the fly, like GtkToolbar or GtkGesturePan.

**glxc.ORIENTATION\_HORIZONTAL** The element is in horizontal orientation.  
**glxc.ORIENTATION\_VERTICAL** The element is in vertical orientation.

## 6.4.7 enum PackType

### **PackType**

Represents the packing location GtkBox children. (See: GtkVBox, GtkHBox, and GtkButtonBox).

**glxc.PACK\_START** The child is packed into the start of the box  
**glxc.PACK\_END** The child is packed into the end of the box

## 6.4.8 enum PositionType

### **PositionType**

Describes which edge of a widget a certain feature is positioned at, e.g. the tabs of a GtkNotebook, the handle of a GtkHandleBox or the label of a GtkScale.

**glxc.POS\_LEFT** The feature is at the left edge.  
**glxc.POS\_RIGHT** The feature is at the right edge.  
**glxc.POS\_TOP** The feature is at the top edge.  
**glxc.POS\_BOTTOM** The feature is at the bottom edge.

### 6.4.9 enum ReliefStyle

#### ReliefStyle

Indicated the relief to be drawn around a GtkButton.

**glxc.RELIEF\_NORMAL** Draw a normal relief.

**glxc.RELIEF\_HALF** A half relief. Deprecated in 3.14, does the same as **glxc.RELIEF\_NORMAL**

**glxc.RELIEF\_NONE** No relief.

### 6.4.10 enum ScrollStep

#### ScrollStep

Type of relief

**glxc.SCROLL\_STEPS** Scroll in steps.

**glxc.SCROLL\_PAGES** Scroll by pages.

**glxc.SCROLL\_ENDS** Scroll to ends.

**glxc.SCROLL\_HORIZONTAL\_STEPS** Scroll in horizontal steps.

**glxc.SCROLL\_HORIZONTAL\_PAGES** Scroll by horizontal pages.

**glxc.SCROLL\_HORIZONTAL\_ENDS** Scroll to the horizontal ends.

### 6.4.11 enum ScrollType

#### ScrollStep

Scrolling types.

**glxc.SCROLL\_NONE** No scrolling.

**glxc.SCROLL\_JUMP** Jump to new location.

**glxc.SCROLL\_STEP\_BACKWARD** Step backward.

**glxc.SCROLL\_STEP\_FORWARD** Step forward.

**glxc.SCROLL\_PAGE\_BACKWARD** Page backward.

**glxc.SCROLL\_PAGE\_FORWARD** Page forward.

**glxc.SCROLL\_STEP\_UP** Step up.

**glxc.SCROLL\_STEP\_DOWN** Step down.

**glxc.SCROLL\_PAGE\_UP** Page up.

**glxc.SCROLL\_PAGE\_DOWN** Page down.

**glxc.SCROLL\_STEP\_LEFT** Step to the left.

**glxc.SCROLL\_STEP\_RIGHT** Step to the right.

**glxc.SCROLL\_PAGE\_LEFT** Page to the left.

**glxc.SCROLL\_PAGE\_RIGHT** Page to the right.

**glxc.SCROLL\_START** Scroll to start.

**glxc.SCROLL\_END** Scroll to end.

### 6.4.12 enum SelectionMode

#### SelectionMode

Used to control what selections users are allowed to make.

**glxc.SELECTION\_NONE** No selection is possible.

**glxc.SELECTION\_SINGLE** Zero or one element may be selected.

**glxc.SELECTION\_BROWSE** Exactly one element is selected. In some circumstances, such as initially or during a search operation, it's possible for no element to be selected with **glxc.SELECTION\_BROWSE**. What is really enforced is that the user can't deselect a currently selected element except by selecting another element.

**glxc.SELECTION\_MULTIPLE** Any number of elements may be selected. The Ctrl key may be used to enlarge the selection, and Shift key to select between the focus and the child pointed to. Some widgets may also allow Click-drag to select a range of elements.

### 6.4.13 enum ShadowType

#### ShadowType

Used to change the appearance of an outline typically provided by a GtkFrame.

Note that many themes do not differentiate the appearance of the various shadow types: Either there is no visible shadow (**glxc.SHADOW\_NONE**), or there is (any other value).

**glxc.SHADOW\_NONE** No outline.

**glxc.SHADOW\_IN** The outline is bevelled inwards.

**glxc.SHADOW\_OUT** The outline is bevelled outwards like a button.

**glxc.SHADOW\_ETCHED\_IN** The outline has a sunken 3d appearance.

**glxc.SHADOW\_ETCHED\_OUT** The outline has a raised 3d appearance.

### 6.4.14 enum StateFlags

#### StateFlags

Describes a widget state. Widget states are used to match the widget against CSS pseudo-classes. Note that GTK extends the regular CSS classes and sometimes uses different names.

**glxc.STATE\_FLAG\_NORMAL** State during normal operation.

**glxc.STATE\_FLAG\_ACTIVE** Widget is active.

**glxc.STATE\_FLAG\_PRELIGHT** Widget has a mouse pointer over it.

**glxc.STATE\_FLAG\_SELECTED** Widget is selected.

**glxc.STATE\_FLAG\_INSENSITIVE** Widget is insensitive.

**glxc.STATE\_FLAG\_INCONSISTENT** Widget is inconsistent.

**glxc.STATE\_FLAG\_FOCUSED** Widget has the keyboard focus.

**glxc.STATE\_FLAG\_BACKDROP** Widget is in a background toplevel window.

**glxc.STATE\_FLAG\_DIR\_LTR** Widget is in left-to-right text direction. Since 3.8

**glxc.STATE\_FLAG\_DIR\_RTL** Widget is in right-to-left text direction. Since 3.8

**glxc.STATE\_FLAG\_LINK** Widget is a link. Since 3.12

**glxc.STATE\_FLAG\_VISITED** The location the widget points to has already been visited.  
Since 3.12

**glxc.STATE\_FLAG\_CHECKED** Widget is checked. Since 3.14

**glxc.STATE\_FLAG\_DROP\_ACTIVE** Widget is highlighted as a drop target for DND.  
Since 3.20

### 6.4.15 enum ToolbarStyle

#### ToolbarStyle

Used to customize the appearance of a GtkToolbar. Note that setting the toolbar style overrides the user's preferences for the default toolbar style. Note that if the button has only a label set and `glxc.TOOLBAR_ICONS` is used, the label will be visible, and vice versa.

**glxc.TOOLBAR\_ICONS** Buttons display only icons in the toolbar.

**glxc.TOOLBAR\_TEXT** Buttons display only text labels in the toolbar.

**glxc.TOOLBAR\_BOTH** Buttons display text and icons in the toolbar.

**glxc.TOOLBAR\_BOTH\_HORIZ** Buttons display icons and text alongside each other,  
rather than vertically stacked

### 6.4.16 enum SortType

#### SortType

Determines the direction of a sort.

**glxc.SORT\_ASCENDING** Sorting is in ascending order.

**glxc.SORT\_DESCENDING** Sorting is in descending order.

## 6.5 Contributors

- Jérôme Ornech alias Tuuux <tuxa at rtnp dot org>
- Aurélien Maury alias M4Momo <mo at rtnp dot org>
- Adam Bouadil alias “Deitsuen” <adamb27750 at orange dot fr>

---

### Note for GTK+ Project Developer's

---

I'm really confuse about the big copy/past i making from the GTK+ documentation during the creation of the Galaxie-Curses documentation, that because english is not my primary language and i'm a bit limited for make a ToolKit documentation without that ... Consider that actual documentation of Galaxie-Curse as the better i can do and it include to copy/past large parts of the GTK+ documentation. (sorry about that)

As you probably see **Galaxie-Curses** is a Text Based **GTK+** like, then the GTK+ Doc is the **roadmap**.





## CHAPTER 8

---

### License

---

See the [LICENCE](#)

All contributions to the project source code (“patches”) SHALL use the same license as the project.



## CHAPTER 9

---

### Indices and tables

---

- `genindex`
- `search`



### g

GLXCurses, 125  
GLXCurses.Actionable, 35  
GLXCurses.Adjustment, 36  
GLXCurses.Aera, 41  
GLXCurses.Application, 42  
GLXCurses.Bin, 44  
GLXCurses.Bindings, 45  
GLXCurses.Box, 45  
GLXCurses.Button, 48  
GLXCurses.Buzzer, 49  
GLXCurses.CheckButton, 50  
GLXCurses.Clipboards, 51  
GLXCurses.Constants, 52  
GLXCurses.Container, 52  
GLXCurses.Dialog, 57  
GLXCurses.Editable, 60  
GLXCurses.Entry, 64  
GLXCurses.EntryBuffer, 78  
GLXCurses.EntryCompletion, 80  
GLXCurses.EventList, 82  
GLXCurses.FileChooser, 82  
GLXCurses.FileChooserMenu, 83  
GLXCurses.Frame, 84  
GLXCurses.HBox, 86  
GLXCurses.HSeparator, 86  
GLXCurses.Image, 87  
GLXCurses.Label, 88  
GLXCurses.libs, 35  
GLXCurses.libs.ApplicationHandlers, 15  
GLXCurses.libs.ChildElement, 15  
GLXCurses.libs.ChildProperty, 16  
GLXCurses.libs.Colorable, 17  
GLXCurses.libs.Colors, 17  
GLXCurses.libs.Dividable, 19  
GLXCurses.libs.File, 19  
GLXCurses.libs.FileChooserFunctions, 20  
GLXCurses.libs.Group, 21  
GLXCurses.libs.GroupElement, 22  
GLXCurses.libs.Groups, 22  
GLXCurses.libs.handlers, 15  
GLXCurses.libs.handlers.application, 14  
GLXCurses.libs.handlers.button, 14  
GLXCurses.libs.handlers.container, 14  
GLXCurses.libs.handlers.editable, 14  
GLXCurses.libs.handlers.filechooser, 14  
GLXCurses.libs.handlers.label, 15  
GLXCurses.libs.handlers.statusbar, 15  
GLXCurses.libs.handlers.textview, 15  
GLXCurses.libs.handlers.widget, 15  
GLXCurses.libs.handlers.window, 15  
GLXCurses.libs.ImageConvert, 23  
GLXCurses.libs.Movable, 24  
GLXCurses.libs.Spot, 24  
GLXCurses.libs.TextAttributes, 28  
GLXCurses.libs.TextFonts, 29  
GLXCurses.libs.TextUtils, 29  
GLXCurses.libs.TTY, 25  
GLXCurses.libs.Utils, 30  
GLXCurses.libs.XDGBaseDirectory, 33  
GLXCurses.MainLoop, 94  
GLXCurses.Menu, 95  
GLXCurses.MenuBar, 95  
GLXCurses.MenuItem, 95  
GLXCurses.MenuShell, 96  
GLXCurses.MessageBar, 96  
GLXCurses.Misc, 98  
GLXCurses.Object, 99  
GLXCurses.ProgressBar, 100  
GLXCurses.RadioButton, 100  
GLXCurses.Range, 100  
GLXCurses.Scrollable, 105  
GLXCurses.StatusBar, 105  
GLXCurses.Style, 107  
GLXCurses.TextBuffer, 108  
GLXCurses.TextTag, 108  
GLXCurses.TextTagTable, 110  
GLXCurses.TextView, 110  
GLXCurses.ToolBar, 112

`GLXCurses.VBox`, [113](#)  
`GLXCurses.VSeparator`, [113](#)  
`GLXCurses.VuMeter`, [113](#)  
`GLXCurses.Widget`, [113](#)  
`GLXCurses.Window`, [120](#)

## A

- `accel_path` (*GLXCurses.MenuItem* attribute), 170
- `accel_path` (*GLXCurses.MenuItem.MenuItem* attribute), 95
- `accelerator_size` (*GLXCurses.MenuItem* attribute), 171
- `accelerator_size` (*GLXCurses.MenuItem.MenuItem* attribute), 96
- `accept_focus` (*GLXCurses.Window* attribute), 152
- `accept_focus` (*GLXCurses.Window.Window* attribute), 120
- `accept_tab` (*GLXCurses.TextView* attribute), 146
- `accept_tab` (*GLXCurses.TextView.TextView* attribute), 110
- `accumulative_margin` (*GLXCurses.TextTag* attribute), 144
- `accumulative_margin` (*GLXCurses.TextTag.TextTag* attribute), 108
- `action_aera_border` (*GLXCurses.Dialog* attribute), 162
- `action_aera_border` (*GLXCurses.Dialog.Dialog* attribute), 57
- `action_name` (*GLXCurses.Actionable* attribute), 209
- `action_name` (*GLXCurses.Actionable.Actionable* attribute), 36
- `action_target` (*GLXCurses.Actionable* attribute), 209
- `action_target` (*GLXCurses.Actionable.Actionable* attribute), 36
- `Actionable` (class in *GLXCurses*), 209
- `Actionable` (class in *GLXCurses.Actionable*), 35
- `activate_default()` (*GLXCurses.Window* method), 156
- `activate_default()` (*GLXCurses.Window.Window* method), 125
- `activate_focus()` (*GLXCurses.Window* method), 156
- `activate_focus()` (*GLXCurses.Window.Window* method), 125
- `activates_default` (*GLXCurses.Entry* attribute), 189
- `activates_default` (*GLXCurses.Entry.Entry* attribute), 64
- `active` (*GLXCurses.CheckButton* attribute), 157
- `active` (*GLXCurses.CheckButton.CheckButton* attribute), 50
- `active` (*GLXCurses.RadioButton* attribute), 157
- `active` (*GLXCurses.RadioButton.RadioButton* attribute), 100
- `active_widgets` (*GLXCurses.libs.Spot.Spot* attribute), 24
- `active_window` (*GLXCurses.Application* attribute), 165
- `active_window` (*GLXCurses.Application.Application* attribute), 42
- `active_window_id` (*GLXCurses.libs.Spot.Spot* attribute), 24
- `active_window_id_prev` (*GLXCurses.libs.Spot.Spot* attribute), 24
- `add()` (*GLXCurses.Container* method), 140
- `add()` (*GLXCurses.Container.Container* method), 53
- `add()` (*GLXCurses.EventList.EventList* method), 82
- `add()` (*GLXCurses.libs.Group.Group* method), 22
- `add()` (*GLXCurses.TextTagTable* method), 146
- `add()` (*GLXCurses.TextTagTable.TextTagTable* method), 110
- `add_accel_group()` (*GLXCurses.Menu* static method), 170
- `add_accel_group()` (*GLXCurses.Menu.Menu* static method), 95
- `add_accel_group()` (*GLXCurses.Window* static method), 156
- `add_accel_group()` (*GLXCurses.Window.Window* static method), 124
- `add_action_widget()` (*GLXCurses.Dialog* method), 164
- `add_action_widget()` (*GLXCurses.Dialog.Dialog* method), 59
- `add_button()` (*GLXCurses.Dialog* method), 163

`add_button()` (*GLXCurses.Dialog.Dialog method*), 58  
`add_buttons()` (*GLXCurses.Dialog method*), 164  
`add_buttons()` (*GLXCurses.Dialog.Dialog method*), 58  
`add_character()` (*GLXCurses.Aera.Area method*), 42  
`add_group()` (*GLXCurses.libs.Groups.Groups method*), 22  
`add_horizontal_line()` (*GLXCurses.Aera.Area method*), 42  
`add_rectangle()` (*GLXCurses.Aera.Area method*), 42  
`add_signal()` (*GLXCurses.Bindings.Binding method*), 45  
`add_string()` (*GLXCurses.Aera.Area method*), 42  
`add_vertical_line()` (*GLXCurses.Aera.Area method*), 42  
`add_window()` (*GLXCurses.Application method*), 166  
`add_window()` (*GLXCurses.Application.Application method*), 43  
`add_with_properties()` (*GLXCurses.Container method*), 140  
`add_with_properties()` (*GLXCurses.Container.Container method*), 53  
`Adjustment` (*class in GLXCurses*), 158  
`Adjustment` (*class in GLXCurses.Adjustment*), 36  
`adjustment` (*GLXCurses.Range attribute*), 204  
`adjustment` (*GLXCurses.Range.Range attribute*), 100  
`angle` (*GLXCurses.Label attribute*), 175  
`angle` (*GLXCurses.Label.Label attribute*), 88  
`app_menu` (*GLXCurses.Application attribute*), 166  
`app_menu` (*GLXCurses.Application.Application attribute*), 43  
`app_paintable` (*GLXCurses.Widget attribute*), 132  
`app_paintable` (*GLXCurses.Widget.Widget attribute*), 113  
`append()` (*GLXCurses.MenuShell method*), 170  
`append()` (*GLXCurses.MenuShell.MenuShell method*), 96  
`Application` (*class in GLXCurses*), 165  
`Application` (*class in GLXCurses.Application*), 42  
`application` (*GLXCurses.MainLoop.MainLoop attribute*), 94  
`application` (*GLXCurses.Window attribute*), 152  
`application` (*GLXCurses.Window.Window attribute*), 120  
`Area` (*class in GLXCurses.Aera*), 41  
`attached_to` (*GLXCurses.Window attribute*), 152  
`attached_to` (*GLXCurses.Window.Window attribute*), 121  
`attribute_states` (*GLXCurses.Widget attribute*), 138  
`attribute_states` (*GLXCurses.Widget attribute*), 119  
`attribute_to_rgb()` (*GLXCurses.Style method*), 131  
`attribute_to_rgb()` (*GLXCurses.Style.Style method*), 107  
`attributes` (*GLXCurses.Entry attribute*), 189  
`attributes` (*GLXCurses.Entry.Entry attribute*), 64  
`attributes` (*GLXCurses.Label attribute*), 175  
`attributes` (*GLXCurses.Label.Label attribute*), 88  
`attributes` (*GLXCurses.libs.TextAttributes.TextAttributes attribute*), 28  
`attributes_states` (*GLXCurses.Style attribute*), 131  
`attributes_states` (*GLXCurses.Style.Style attribute*), 107

## B

`background` (*GLXCurses.TextTag attribute*), 144  
`background` (*GLXCurses.TextTag.TextTag attribute*), 108  
`background_color_normal` (*GLXCurses.libs.Colorable.Colorable attribute*), 17  
`background_color_prelight` (*GLXCurses.libs.Colorable.Colorable attribute*), 17  
`background_full_height` (*GLXCurses.TextTag attribute*), 144  
`background_full_height` (*GLXCurses.TextTag.TextTag attribute*), 109  
`background_full_height_set` (*GLXCurses.TextTag attribute*), 144  
`background_full_height_set` (*GLXCurses.TextTag.TextTag attribute*), 109  
`background_rgb` (*GLXCurses.TextTag attribute*), 145  
`background_rgb` (*GLXCurses.TextTag.TextTag attribute*), 109  
`background_set` (*GLXCurses.TextTag attribute*), 145  
`background_set` (*GLXCurses.TextTag.TextTag attribute*), 109  
`baseline_position` (*GLXCurses.Box attribute*), 148  
`baseline_position` (*GLXCurses.Box.Box attribute*), 45  
`BaselinePosition` (*built-in variable*), 213  
`Bin` (*class in GLXCurses*), 148  
`Bin` (*class in GLXCurses.Bin*), 44  
`Binding` (*class in GLXCurses.Bindings*), 45  
`bool` (*built-in variable*), 213  
`border_width` (*GLXCurses.Container attribute*), 139  
`border_width` (*GLXCurses.Container.Container attribute*), 52  
`bottom_margin` (*GLXCurses.TextView attribute*), 146



- bottom\_margin (*GLXCurses.TextView.TextView attribute*), 110
- Box (*class in GLXCurses*), 148
- Box (*class in GLXCurses.Box*), 45
- buffer (*GLXCurses.Entry attribute*), 189
- buffer (*GLXCurses.Entry.Entry attribute*), 64
- buffer (*GLXCurses.EventList.EventList attribute*), 82
- buffer (*GLXCurses.TextView attribute*), 146
- buffer (*GLXCurses.TextView.TextView attribute*), 110
- Button (*class in GLXCurses.Button*), 48
- button\_spacing (*GLXCurses.Dialog attribute*), 162
- button\_spacing (*GLXCurses.Dialog.Dialog attribute*), 57
- Buzzer (*class in GLXCurses.Buzzer*), 49
- ## C
- cache\_path (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory attribute*), 35
- can\_default (*GLXCurses.Widget attribute*), 132
- can\_default (*GLXCurses.Widget.Widget attribute*), 114
- can\_focus (*GLXCurses.Widget attribute*), 133
- can\_focus (*GLXCurses.Widget.Widget attribute*), 114
- can\_prelight (*GLXCurses.Widget attribute*), 133
- can\_prelight (*GLXCurses.Widget.Widget attribute*), 114
- caps\_locks\_warning (*GLXCurses.Entry attribute*), 189
- caps\_locks\_warning (*GLXCurses.Entry.Entry attribute*), 65
- category (*GLXCurses.libs.FileChooserFunctions.FileChooserFunctions attribute*), 20
- cbreak (*GLXCurses.libs.TTY.Screen attribute*), 25
- cbreak (*GLXCurses.Screen attribute*), 126
- check\_justification() (*GLXCurses.libs.Movable.Movable method*), 24
- check\_mnemonic\_in\_text() (*in module GLXCurses.libs.Utils*), 30
- check\_position() (*GLXCurses.libs.Movable.Movable method*), 24
- check\_resize() (*GLXCurses.Container method*), 141
- check\_resize() (*GLXCurses.Container.Container method*), 54
- check\_sizes() (*GLXCurses.Application method*), 167
- check\_sizes() (*GLXCurses.Application.Application method*), 44
- check\_terminal() (*GLXCurses.libs.TTY.Screen static method*), 28
- check\_terminal() (*GLXCurses.Screen static method*), 129
- CheckBox (*class in GLXCurses*), 157
- CheckBox (*class in GLXCurses.CheckBox*), 50
- child (*GLXCurses.Container attribute*), 139
- child (*GLXCurses.Container.Container attribute*), 53
- child\_get() (*GLXCurses.Container method*), 143
- child\_get() (*GLXCurses.Container.Container method*), 56
- child\_get\_property() (*GLXCurses.Container method*), 143
- child\_get\_property() (*GLXCurses.Container.Container method*), 56
- child\_set() (*GLXCurses.Container method*), 142
- child\_set() (*GLXCurses.Container.Container method*), 56
- child\_set\_property() (*GLXCurses.Container method*), 143
- child\_set\_property() (*GLXCurses.Container.Container method*), 56
- child\_type() (*GLXCurses.Container method*), 142
- child\_type() (*GLXCurses.Container.Container method*), 56
- child\_visible (*GLXCurses.Widget attribute*), 138
- child\_visible (*GLXCurses.Widget.Widget attribute*), 119
- ChildElement (*class in GLXCurses.libs.ChildElement*), 15
- ChildProperty (*class in GLXCurses.libs.ChildProperty*), 16
- children (*GLXCurses.Application attribute*), 166
- children (*GLXCurses.Application.Application attribute*), 43
- children (*GLXCurses.Object attribute*), 132
- children (*GLXCurses.Object.Object attribute*), 99
- clamp() (*in module GLXCurses.libs.Utils*), 32
- clamp\_page() (*GLXCurses.Adjustment method*), 160
- clamp\_page() (*GLXCurses.Adjustment.Adjustment method*), 38
- clamp\_to\_zero() (*in module GLXCurses.libs.Utils*), 32
- Clipboard (*class in GLXCurses*), 125
- Clipboard (*class in GLXCurses.Clipboards*), 51
- close() (*GLXCurses.Dialog method*), 165
- close() (*GLXCurses.Dialog.Dialog method*), 60
- close() (*GLXCurses.libs.TTY.Screen method*), 27
- close() (*GLXCurses.Screen method*), 129
- color (*GLXCurses.Button.Button attribute*), 49
- color (*GLXCurses.CheckBox attribute*), 158
- color (*GLXCurses.CheckBox.CheckBox attribute*), 51
- color (*GLXCurses.Menu attribute*), 170
- color (*GLXCurses.Menu.Menu attribute*), 95
- color (*GLXCurses.MenuBar attribute*), 170
- color (*GLXCurses.MenuBar.MenuBar attribute*), 95
- color (*GLXCurses.MenuItem attribute*), 171
- color (*GLXCurses.MenuItem.MenuItem attribute*), 96
- color (*GLXCurses.RadioButton attribute*), 157

- color (*GLXCurses.RadioButton.RadioButton attribute*), 100  
 color (*GLXCurses.Window attribute*), 156  
 color (*GLXCurses.Window.Window attribute*), 124  
 color() (*GLXCurses.Colors method*), 130  
 color() (*GLXCurses.libs.Colors.Colors method*), 18  
 color() (*GLXCurses.ProgressBar method*), 182  
 color() (*GLXCurses.ProgressBar.ProgressBar method*), 100  
 color\_detection\_value (*GLXCurses.Colors attribute*), 130  
 color\_detection\_value (*GLXCurses.libs.Colors.Colors attribute*), 18  
 color\_insensitive (*GLXCurses.libs.Colorable.Colorable attribute*), 17  
 color\_normal (*GLXCurses.libs.Colorable.Colorable attribute*), 17  
 color\_prelight (*GLXCurses.libs.Colorable.Colorable attribute*), 17  
 Colorable (*class in GLXCurses.libs.Colorable*), 17  
 Colors (*class in GLXCurses*), 129  
 Colors (*class in GLXCurses.libs.Colors*), 17  
 completion (*GLXCurses.Entry attribute*), 190  
 completion (*GLXCurses.Entry.Entry attribute*), 65  
 composite\_child (*GLXCurses.Widget attribute*), 133  
 composite\_child (*GLXCurses.Widget.Widget attribute*), 114  
 config\_path (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory attribute*), 35  
 config\_paths (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory attribute*), 35  
 configure() (*GLXCurses.Adjustment method*), 160  
 configure() (*GLXCurses.Adjustment.Adjustment method*), 39  
 Constants (*class in GLXCurses.Constants*), 52  
 Constants.ConstError, 52  
 Container (*class in GLXCurses*), 139  
 Container (*class in GLXCurses.Container*), 52  
 content\_area\_border (*GLXCurses.Dialog attribute*), 163  
 content\_area\_border (*GLXCurses.Dialog.Dialog attribute*), 57  
 content\_area\_spacing (*GLXCurses.Dialog attribute*), 163  
 content\_area\_spacing (*GLXCurses.Dialog.Dialog attribute*), 57  
 control\_directory() (*in module GLXCurses.libs.XDGBaseDirectory*), 33  
 convert\_file\_attribute() (*GLXCurses.libs.FileChooserFunctions.FileChooserUtils method*), 21  
 copy\_clipboard() (*GLXCurses.Editable method*), 187  
 copy\_clipboard() (*GLXCurses.Editable.Editable method*), 62  
 curses\_color() (*GLXCurses.Colors static method*), 130  
 curses\_color() (*GLXCurses.libs.Colors.Colors static method*), 18  
 curses\_color\_pair\_number() (*GLXCurses.Colors static method*), 130  
 curses\_color\_pair\_number() (*GLXCurses.libs.Colors.Colors static method*), 18  
 curses\_color\_pairs\_init() (*GLXCurses.Colors method*), 130  
 curses\_color\_pairs\_init() (*GLXCurses.libs.Colors.Colors method*), 18  
 cursor\_position (*GLXCurses.Entry attribute*), 190  
 cursor\_position (*GLXCurses.Entry.Entry attribute*), 65  
 cursor\_position (*GLXCurses.Label attribute*), 176  
 cursor\_position (*GLXCurses.Label.Label attribute*), 88  
 cursor\_position (*GLXCurses.TextBuffer attribute*), 146  
 cursor\_position (*GLXCurses.TextBuffer.TextBuffer attribute*), 108  
 cursor\_visible (*GLXCurses.TextView attribute*), 146  
 cursor\_visible (*GLXCurses.TextView.TextView attribute*), 111  
 cut\_clipboard() (*GLXCurses.Editable method*), 187  
 cut\_clipboard() (*GLXCurses.Editable.Editable method*), 62  
 cwd (*GLXCurses.libs.FileChooserFunctions.FileChooserUtils attribute*), 20
- ## D
- data (*GLXCurses.Image attribute*), 211  
 data (*GLXCurses.Image.Image attribute*), 87  
 data (*GLXCurses.ImageConvert attribute*), 212  
 data (*GLXCurses.libs.ImageConvert.ImageConvert attribute*), 23  
 data\_path (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory attribute*), 35  
 data\_paths (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory attribute*), 35  
 debug (*GLXCurses.EventList.EventList attribute*), 82  
 debug (*GLXCurses.MainLoop.MainLoop attribute*), 94  
 debug (*GLXCurses.Object attribute*), 132  
 debug (*GLXCurses.Object.Object attribute*), 99

- debug\_level (*GLXCurses.Object* attribute), 132  
 debug\_level (*GLXCurses.Object.Object* attribute), 99  
 decorated (*GLXCurses.Window* attribute), 153  
 decorated (*GLXCurses.Window.Window* attribute), 121  
 decoration\_button\_layout (*GLXCurses.Window* attribute), 156  
 decoration\_button\_layout (*GLXCurses.Window.Window* attribute), 124  
 decoration\_resize\_handle (*GLXCurses.Window* attribute), 156  
 decoration\_resize\_handle (*GLXCurses.Window.Window* attribute), 124  
 default\_attributes\_states (*GLXCurses.Style* attribute), 131  
 default\_attributes\_states (*GLXCurses.Style.Style* attribute), 107  
 default\_flags (*GLXCurses.Object* attribute), 132  
 default\_flags (*GLXCurses.Object.Object* attribute), 99  
 default\_height (*GLXCurses.Window* attribute), 153  
 default\_height (*GLXCurses.Window.Window* attribute), 121  
 default\_width (*GLXCurses.Window* attribute), 153  
 default\_width (*GLXCurses.Window.Window* attribute), 121  
 deletable (*GLXCurses.Window* attribute), 153  
 deletable (*GLXCurses.Window.Window* attribute), 121  
 delete\_selection() (*GLXCurses.Editable* method), 187  
 delete\_selection() (*GLXCurses.Editable.Editable* method), 62  
 delete\_text() (*GLXCurses.Editable* method), 186  
 delete\_text() (*GLXCurses.Editable.Editable* method), 61  
 delete\_text() (*GLXCurses.EntryBuffer* method), 185  
 delete\_text() (*GLXCurses.EntryBuffer.EntryBuffer* method), 80  
 DeleteType (built-in variable), 214  
 destroy() (*GLXCurses.Object* method), 132  
 destroy() (*GLXCurses.Object.Object* method), 99  
 destroy() (*GLXCurses.Widget* method), 137  
 destroy() (*GLXCurses.Widget.Widget* method), 118  
 destroy\_with\_parent (*GLXCurses.Window* attribute), 153  
 destroy\_with\_parent (*GLXCurses.Window.Window* attribute), 121  
 destroyed() (*GLXCurses.Widget* method), 137  
 destroyed() (*GLXCurses.Widget.Widget* method), 118  
 Dialog (class in *GLXCurses.Dialog*), 57  
 dict (built-in variable), 213  
 direction (*GLXCurses.TextTag* attribute), 145  
 direction (*GLXCurses.TextTag.TextTag* attribute), 109  
 DirectionType (built-in variable), 214  
 directory (*GLXCurses.libs.File.File* attribute), 19  
 directory\_view (*GLXCurses.libs.FileChooserFunctions.FileChooserUtils* attribute), 20  
 disk\_usage() (in module *GLXCurses.libs.Utils*), 31  
 Dividable (class in *GLXCurses.libs.Dividable*), 19  
 down() (*GLXCurses.libs.Group.Group* method), 22  
 down() (*GLXCurses.libs.Groups.Groups* method), 23  
 draw() (*GLXCurses.Menuitem* method), 171  
 draw() (*GLXCurses.Menuitem.Menuitem* method), 96  
 draw() (*GLXCurses.MessageBar* method), 173  
 draw() (*GLXCurses.MessageBar.MessageBar* method), 98  
 draw() (*GLXCurses.StatusBar* method), 172  
 draw() (*GLXCurses.StatusBar.StatusBar* method), 107  
 draw() (*GLXCurses.ToolBar* method), 174  
 draw() (*GLXCurses.ToolBar.ToolBar* method), 112  
 draw() (*GLXCurses.Widget* method), 139  
 draw() (*GLXCurses.Widget.Widget* method), 120  
 draw\_background() (*GLXCurses.Aera.Area* method), 42  
 draw\_titles() (*GLXCurses.FileChooserMenu.FileChooserMenu* method), 84  
 draw\_widget\_in\_area() (*GLXCurses.Button.Button* method), 49  
 draw\_widget\_in\_area() (*GLXCurses.CheckButton* method), 158  
 draw\_widget\_in\_area() (*GLXCurses.CheckButton.CheckButton* method), 51  
 draw\_widget\_in\_area() (*GLXCurses.Dialog* method), 165  
 draw\_widget\_in\_area() (*GLXCurses.Dialog.Dialog* method), 60  
 draw\_widget\_in\_area() (*GLXCurses.Entry* method), 194  
 draw\_widget\_in\_area() (*GLXCurses.Entry.Entry* method), 69  
 draw\_widget\_in\_area() (*GLXCurses.FileChooser.FileSelect* method), 83  
 draw\_widget\_in\_area() (*GLXCurses.FileChooserMenu.FileChooserMenu* method), 83  
 draw\_widget\_in\_area() (*GLXCurses.FileSelect* method), 211  
 draw\_widget\_in\_area() (*GLXCurses.Frame*

method), 169  
draw\_widget\_in\_area() (GLX-Curses.Frame.Frame method), 86  
draw\_widget\_in\_area() (GLXCurses.HBox method), 152  
draw\_widget\_in\_area() (GLXCurses.HBox.HBox method), 86  
draw\_widget\_in\_area() (GLXCurses.HSeparator method), 182  
draw\_widget\_in\_area() (GLX-Curses.HSeparator.HSeparator method), 86  
draw\_widget\_in\_area() (GLXCurses.Image method), 212  
draw\_widget\_in\_area() (GLX-Curses.Image.Image method), 88  
draw\_widget\_in\_area() (GLXCurses.Label method), 181  
draw\_widget\_in\_area() (GLXCurses.Label.Label method), 93  
draw\_widget\_in\_area() (GLXCurses.Menu method), 170  
draw\_widget\_in\_area() (GLXCurses.Menu.Menu method), 95  
draw\_widget\_in\_area() (GLXCurses.MenuBar method), 170  
draw\_widget\_in\_area() (GLX-Curses.MenuBar.MenuBar method), 95  
draw\_widget\_in\_area() (GLX-Curses.ProgressBar method), 182  
draw\_widget\_in\_area() (GLX-Curses.ProgressBar.ProgressBar method), 100  
draw\_widget\_in\_area() (GLX-Curses.RadioButton method), 157  
draw\_widget\_in\_area() (GLX-Curses.RadioButton.RadioButton method), 100  
draw\_widget\_in\_area() (GLXCurses.VBox method), 151  
draw\_widget\_in\_area() (GLXCurses.VBox.VBox method), 113  
draw\_widget\_in\_area() (GLXCurses.VSeparator method), 182  
draw\_widget\_in\_area() (GLX-Curses.VSeparator.VSeparator method), 113  
draw\_widget\_in\_area() (GLXCurses.Widget method), 139  
draw\_widget\_in\_area() (GLX-Curses.Widget.Widget method), 120  
draw\_widget\_in\_area() (GLXCurses.Window method), 156  
draw\_widget\_in\_area() (GLX-

Curses.Window.Window method), 124

## E

echo (GLXCurses.lib.TTY.Screen attribute), 25  
echo (GLXCurses.Screen attribute), 127  
Editable (class in GLXCurses), 185  
Editable (class in GLXCurses.Editable), 60  
editable (GLXCurses.Entry attribute), 190  
editable (GLXCurses.Entry.Entry attribute), 65  
editable (GLXCurses.TextTag attribute), 145  
editable (GLXCurses.TextTag.TextTag attribute), 109  
editable (GLXCurses.TextView attribute), 147  
editable (GLXCurses.TextView.TextView attribute), 111  
editable\_set (GLXCurses.TextTag attribute), 145  
editable\_set (GLXCurses.TextTag.TextTag attribute), 109  
emit\_changed() (GLXCurses.Adjustment method), 160  
emit\_changed() (GLX-Curses.Adjustment.Adjustment method), 39  
emit\_value\_changed() (GLXCurses.Adjustment method), 160  
emit\_value\_changed() (GLX-Curses.Adjustment.Adjustment method), 39  
enable\_emoji\_completion (GLXCurses.Entry attribute), 190  
enable\_emoji\_completion (GLX-Curses.Entry.Entry attribute), 65  
Entry (class in GLXCurses), 189  
Entry (class in GLXCurses.Entry), 64  
EntryBuffer (class in GLXCurses), 182  
EntryBuffer (class in GLXCurses.EntryBuffer), 78  
EntryCompletion (class in GLXCurses), 202  
EntryCompletion (class in GLX-Curses.EntryCompletion), 80  
eveloop\_cmd() (GLXCurses.Application method), 167  
eveloop\_cmd() (GLXCurses.Application.Application method), 44  
eveloop\_dispatch() (GLXCurses.Object method), 132  
eveloop\_dispatch() (GLXCurses.Object.Object method), 99  
eveloop\_dispatch\_application() (GLX-Curses.Application method), 167  
eveloop\_dispatch\_application() (GLX-Curses.Application.Application method), 44  
eveloop\_finalization() (GLX-Curses.Application method), 167



- eveloop\_finalization() (GLXCurses.Application.Application method), 44  
 eveloop\_input\_event() (GLXCurses.Application method), 167  
 eveloop\_input\_event() (GLXCurses.Application.Application method), 44  
 eveloop\_keyboard\_interruption() (GLXCurses.Application method), 167  
 eveloop\_keyboard\_interruption() (GLXCurses.Application.Application method), 44  
 event\_list (GLXCurses.MainLoop.MainLoop attribute), 94  
 EventList (class in GLXCurses.EventList), 82  
 events (GLXCurses.Widget attribute), 133  
 events (GLXCurses.Widget.Widget attribute), 114  
 expand (GLXCurses.libs.ChildProperty.ChildProperty attribute), 16  
 expand (GLXCurses.Widget attribute), 133  
 expand (GLXCurses.Widget.Widget attribute), 114  
 extension (GLXCurses.libs.File.File attribute), 19  
 extension (GLXCurses.libs.FileChooserFunctions.FileChooserUtils attribute), 20
- ## F
- family (GLXCurses.TextTag attribute), 145  
 family (GLXCurses.TextTag.TextTag attribute), 109  
 family\_set (GLXCurses.TextTag attribute), 145  
 family\_set (GLXCurses.TextTag.TextTag attribute), 110  
 File (class in GLXCurses.libs.File), 19  
 FILE\_HIGH\_LIGHT (GLXCurses.libs.FileChooserFunctions.FileChooserUtils attribute), 20  
 FILE\_HIGH\_LIGHT\_PREP (GLXCurses.libs.FileChooserFunctions.FileChooserUtils attribute), 20  
 FileChooserMenu (class in GLXCurses.FileChooserMenu), 83  
 FileChooserUtils (class in GLXCurses.libs.FileChooserFunctions), 20  
 FileSelect (class in GLXCurses), 210  
 FileSelect (class in GLXCurses.FileChooser), 82  
 fill (GLXCurses.libs.ChildProperty.ChildProperty attribute), 16  
 fill\_level (GLXCurses.Range attribute), 204  
 fill\_level (GLXCurses.Range.Range attribute), 100  
 flags (GLXCurses.Object attribute), 132  
 flags (GLXCurses.Object.Object attribute), 99  
 float (built-in variable), 213  
 focus\_on\_click (GLXCurses.Widget attribute), 133  
 focus\_on\_click (GLXCurses.Widget.Widget attribute), 114  
 focus\_on\_map (GLXCurses.Window attribute), 153  
 focus\_on\_map (GLXCurses.Window.Window attribute), 121  
 focus\_visible (GLXCurses.Window attribute), 153  
 focus\_visible (GLXCurses.Window.Window attribute), 121  
 forall() (GLXCurses.Container method), 141  
 forall() (GLXCurses.Container.Container method), 55  
 foreach() (GLXCurses.TextTagTable method), 146  
 foreach() (GLXCurses.TextTagTable.TextTagTable method), 110  
 foreachs() (GLXCurses.Container method), 141  
 foreachs() (GLXCurses.Container.Container method), 54  
 foreground\_color\_normal (GLXCurses.libs.Colorable.Colorable attribute), 17  
 foreground\_color\_prelight (GLXCurses.libs.Colorable.Colorable attribute), 17  
 foreground\_color\_prelight (GLXCurses.libs.Colorable.Colorable attribute), 17  
 foreground\_output\_file\_name() (GLXCurses.libs.File.File method), 20  
 Frame (class in GLXCurses), 167  
 Frame (class in GLXCurses.Frame), 84
- ## G
- get() (GLXCurses.Clipboard method), 125  
 get() (GLXCurses.Clipboards.Clipboard method), 51  
 get\_action\_area() (GLXCurses.Dialog method), 165  
 get\_action\_area() (GLXCurses.Dialog.Dialog method), 60  
 get\_action\_name() (GLXCurses.Actionable method), 209  
 get\_action\_name() (GLXCurses.Actionable.Actionable method), 36  
 get\_action\_target\_value() (GLXCurses.Actionable method), 209  
 get\_action\_target\_value() (GLXCurses.Actionable.Actionable method), 36  
 get\_activates\_default() (GLXCurses.Entry method), 196  
 get\_activates\_default() (GLXCurses.Entry.Entry method), 71  
 get\_adjustment() (GLXCurses.Range method), 206  
 get\_adjustment() (GLXCurses.Range.Range method), 102  
 get\_alignment() (GLXCurses.Entry method), 198  
 get\_alignment() (GLXCurses.Entry.Entry method), 73

`get_app_file_extensions()` (GLXCurses.libs.FileChooserFunctions.FileChooserUtils method), 21

`get_app_file_extensions()` (GLXCurses.libs.FileChooserFunctions.FileChooserUtils method), 173

`get_attributes()` (GLXCurses.Entry method), 199

`get_attributes()` (GLXCurses.Entry.Entry method), 74

`get_attributes_by_filename()` (GLXCurses.libs.FileChooserFunctions.FileChooserUtils method), 21

`get_attributes_by_filename()` (GLXCurses.libs.FileChooserFunctions.FileChooserUtils method), 173

`get_blanche()` (GLXCurses.Buzzer.Buzzer method), 50

`get_border_width()` (GLXCurses.Container method), 143

`get_border_width()` (GLXCurses.Container.Container method), 57

`get_buffer()` (GLXCurses.Entry method), 194

`get_buffer()` (GLXCurses.Entry.Entry method), 69

`get_button_by_x_coord()` (GLXCurses.ToolBar method), 174

`get_button_by_x_coord()` (GLXCurses.ToolBar.ToolBar method), 112

`get_button_width()` (GLXCurses.ToolBar method), 174

`get_button_width()` (GLXCurses.ToolBar.ToolBar method), 112

`get_bytes()` (GLXCurses.EntryBuffer method), 183

`get_bytes()` (GLXCurses.EntryBuffer.EntryBuffer method), 79

`get_center_widget()` (GLXCurses.Box method), 151

`get_center_widget()` (GLXCurses.Box.Box method), 48

`get_chars()` (GLXCurses.Editable method), 186

`get_chars()` (GLXCurses.Editable.Editable method), 62

`get_child()` (GLXCurses.Bin method), 148

`get_child()` (GLXCurses.Bin.Bin method), 45

`get_child_x_coordinates()` (GLXCurses.libs.Dividable.Dividable static method), 19

`get_child_y_coordinates()` (GLXCurses.libs.Dividable.Dividable static method), 19

`get_color_by_filename()` (GLXCurses.libs.FileChooserFunctions.FileChooserUtils method), 21

`get_color_by_filename()` (GLXCurses.libs.FileChooserFunctions.FileChooserUtils method), 173

`get_completion()` (GLXCurses.Entry method), 199

`get_completion()` (GLXCurses.Entry.Entry method), 74

`get_content_area()` (GLXCurses.Dialog method), 165

`get_content_area()` (GLXCurses.Dialog.Dialog method), 60

`get_context_id()` (GLXCurses.MessageBar method), 173

`get_context_id()` (GLXCurses.MessageBar.MessageBar method), 97

`get_context_id()` (GLXCurses.StatusBar method), 171

`get_context_id()` (GLXCurses.StatusBar.StatusBar method), 106

`get_croche()` (GLXCurses.Buzzer.Buzzer method), 49

`get_current_icon_drag_source()` (GLXCurses.Entry method), 202

`get_current_icon_drag_source()` (GLXCurses.Entry.Entry method), 77

`get_cursor_hadjustment()` (GLXCurses.Entry method), 199

`get_cursor_hadjustment()` (GLXCurses.Entry.Entry method), 75

`get_decorated()` (GLXCurses.Widget method), 138

`get_decorated()` (GLXCurses.Widget.Widget method), 119

`get_default_widget()` (GLXCurses.Window method), 157

`get_default_widget()` (GLXCurses.Window.Window method), 125

`get_double_croche()` (GLXCurses.Buzzer.Buzzer method), 49

`get_editable()` (GLXCurses.Editable method), 188

`get_editable()` (GLXCurses.Editable.Editable method), 64

`get_fill_level()` (GLXCurses.Range method), 205

`get_fill_level()` (GLXCurses.Range.Range method), 101

`get_flippable()` (GLXCurses.Range method), 208

`get_flippable()` (GLXCurses.Range.Range method), 104

`get_focus()` (GLXCurses.Window method), 157

`get_focus()` (GLXCurses.Window.Window method), 125

`get_focus_chain()` (GLXCurses.Container method), 141

`get_focus_chain()` (GLXCurses.Container.Container method), 55

`get_focus_child()` (GLXCurses.Container method), 142

`get_focus_child()` (GLXCurses.Container.Container method), 55

`get_focus_hadjustment()` (GLXCurses.Container method), 142

`get_focus_hadjustment()` (GLXCurses.Container.Container method), 55

`get_focus_vadjustment()` (GLXCurses.Container method), 142

[get\\_focus\\_vadjustment\(\)](#) (*GLXCurses.Container.Container method*), 55  
[get\\_has\\_frame\(\)](#) (*GLXCurses.Entry method*), 196  
[get\\_has\\_frame\(\)](#) (*GLXCurses.Entry.Entry method*), 71  
[get\\_hertz\\_to\\_ms\(\)](#) (*GLXCurses.Buzzer.Buzzer static method*), 50  
[get\\_icon\\_activatable\(\)](#) (*GLXCurses.Entry method*), 201  
[get\\_icon\\_activatable\(\)](#) (*GLXCurses.Entry.Entry method*), 76  
[get\\_icon\\_area\(\)](#) (*GLXCurses.Entry method*), 202  
[get\\_icon\\_area\(\)](#) (*GLXCurses.Entry.Entry method*), 77  
[get\\_icon\\_at\\_pos\(\)](#) (*GLXCurses.Entry method*), 201  
[get\\_icon\\_at\\_pos\(\)](#) (*GLXCurses.Entry.Entry method*), 77  
[get\\_icon\\_gicon\(\)](#) (*GLXCurses.Entry method*), 201  
[get\\_icon\\_gicon\(\)](#) (*GLXCurses.Entry.Entry method*), 76  
[get\\_icon\\_name\(\)](#) (*GLXCurses.Entry method*), 201  
[get\\_icon\\_name\(\)](#) (*GLXCurses.Entry.Entry method*), 76  
[get\\_icon\\_pixbuf\(\)](#) (*GLXCurses.Entry method*), 201  
[get\\_icon\\_pixbuf\(\)](#) (*GLXCurses.Entry.Entry method*), 76  
[get\\_icon\\_sensitive\(\)](#) (*GLXCurses.Entry method*), 201  
[get\\_icon\\_sensitive\(\)](#) (*GLXCurses.Entry.Entry method*), 77  
[get\\_icon\\_stock\(\)](#) (*GLXCurses.Entry method*), 201  
[get\\_icon\\_stock\(\)](#) (*GLXCurses.Entry.Entry method*), 76  
[get\\_icon\\_storage\\_type\(\)](#) (*GLXCurses.Entry method*), 201  
[get\\_icon\\_storage\\_type\(\)](#) (*GLXCurses.Entry.Entry method*), 76  
[get\\_icon\\_tooltip\\_markup\(\)](#) (*GLXCurses.Entry method*), 202  
[get\\_icon\\_tooltip\\_markup\(\)](#) (*GLXCurses.Entry.Entry method*), 77  
[get\\_icon\\_tooltip\\_text\(\)](#) (*GLXCurses.Entry method*), 202  
[get\\_icon\\_tooltip\\_text\(\)](#) (*GLXCurses.Entry.Entry method*), 77  
[get\\_infos\\_by\\_filename\(\)](#) (*GLXCurses.libs.FileChooserFunctions.FileChooserUtils method*), 21  
[get\\_inner\\_border\(\)](#) (*GLXCurses.Entry method*), 196  
[get\\_inner\\_border\(\)](#) (*GLXCurses.Entry.Entry method*), 71  
[get\\_input\\_hints\(\)](#) (*GLXCurses.Entry method*), 202  
[get\\_input\\_hints\(\)](#) (*GLXCurses.Entry.Entry method*), 77  
[get\\_input\\_purpose\(\)](#) (*GLXCurses.Entry method*), 202  
[get\\_input\\_purpose\(\)](#) (*GLXCurses.Entry.Entry method*), 77  
[get\\_inverted\(\)](#) (*GLXCurses.Range method*), 206  
[get\\_inverted\(\)](#) (*GLXCurses.Range.Range method*), 103  
[get\\_invisible\\_char\(\)](#) (*GLXCurses.Entry method*), 197  
[get\\_invisible\\_char\(\)](#) (*GLXCurses.Entry.Entry method*), 73  
[get\\_justify\(\)](#) (*GLXCurses.Label method*), 181  
[get\\_justify\(\)](#) (*GLXCurses.Label.Label method*), 93  
[get\\_label\(\)](#) (*GLXCurses.Frame method*), 169  
[get\\_label\(\)](#) (*GLXCurses.Frame.Frame method*), 85  
[get\\_label\\_align\(\)](#) (*GLXCurses.Frame method*), 169  
[get\\_label\\_align\(\)](#) (*GLXCurses.Frame.Frame method*), 85  
[get\\_label\\_widget\(\)](#) (*GLXCurses.Frame method*), 169  
[get\\_label\\_widget\(\)](#) (*GLXCurses.Frame.Frame method*), 85  
[get\\_layout\(\)](#) (*GLXCurses.Entry method*), 198  
[get\\_layout\(\)](#) (*GLXCurses.Entry.Entry method*), 74  
[get\\_layout\\_offsets\(\)](#) (*GLXCurses.Entry method*), 198  
[get\\_layout\\_offsets\(\)](#) (*GLXCurses.Entry.Entry method*), 74  
[get\\_length\(\)](#) (*GLXCurses.EntryBuffer method*), 184  
[get\\_length\(\)](#) (*GLXCurses.EntryBuffer.EntryBuffer method*), 79  
[get\\_line\\_wrap\(\)](#) (*GLXCurses.Label method*), 181  
[get\\_line\\_wrap\(\)](#) (*GLXCurses.Label.Label method*), 94  
[get\\_line\\_wrap\\_mode\(\)](#) (*GLXCurses.Label method*), 181  
[get\\_line\\_wrap\\_mode\(\)](#) (*GLXCurses.Label.Label method*), 94  
[get\\_lower\(\)](#) (*GLXCurses.Adjustment method*), 160  
[get\\_lower\(\)](#) (*GLXCurses.Adjustment.Adjustment method*), 39  
[get\\_lower\\_stepper\\_sensitivity\(\)](#) (*GLXCurses.Range method*), 207  
[get\\_lower\\_stepper\\_sensitivity\(\)](#) (*GLXCurses.Range.Range method*), 104  
[get\\_luma\\_component\\_rgb\(\)](#) (*GLXCurses.Colors method*), 130  
[get\\_luma\\_component\\_rgb\(\)](#) (*GLXCurses.libs.Colors.Colors method*), 18

`get_max_length()` (*GLXCurses.Entry* method), 199  
`get_max_length()` (*GLXCurses.Entry.Entry* method), 74  
`get_max_length()` (*GLXCurses.EntryBuffer* method), 184  
`get_max_length()` (*GLXCurses.EntryBuffer.EntryBuffer* method), 79  
`get_max_width_chars()` (*GLXCurses.Entry* method), 196  
`get_max_width_chars()` (*GLXCurses.Entry.Entry* method), 71  
`get_max_width_chars()` (*GLXCurses.Label* method), 181  
`get_max_width_chars()` (*GLXCurses.Label.Label* method), 94  
`get_minimum_increment()` (*GLXCurses.Adjustment* method), 161  
`get_minimum_increment()` (*GLXCurses.Adjustment.Adjustment* method), 40  
`get_mnemonic_keyval()` (*GLXCurses.Label* method), 180  
`get_mnemonic_keyval()` (*GLXCurses.Label.Label* method), 92  
`get_mnemonic_widget()` (*GLXCurses.Label* method), 181  
`get_mnemonic_widget()` (*GLXCurses.Label.Label* method), 93  
`get_mouse()` (*GLXCurses.Application* method), 167  
`get_mouse()` (*GLXCurses.Application.Application* method), 44  
`get_mouse()` (*GLXCurses.libs.TTY.Screen* static method), 28  
`get_mouse()` (*GLXCurses.Screen* static method), 129  
`get_ms_to_tempo()` (*GLXCurses.Buzzer.Buzzer* static method), 50  
`get_notes()` (*GLXCurses.Buzzer.Buzzer* method), 50  
`get_os_temporary_dir()` (in module *GLXCurses.libs.Util*s), 33  
`get_overwrite_mode()` (*GLXCurses.Entry* method), 198  
`get_overwrite_mode()` (*GLXCurses.Entry.Entry* method), 73  
`get_page_increment()` (*GLXCurses.Adjustment* method), 161  
`get_page_increment()` (*GLXCurses.Adjustment.Adjustment* method), 39  
`get_page_size()` (*GLXCurses.Adjustment* method), 161  
`get_page_size()` (*GLXCurses.Adjustment.Adjustment* method), 39  
`get_path_for_child()` (*GLXCurses.Container* method), 141  
`get_path_for_child()` (*GLXCurses.Container.Container* method), 55  
`get_placeholder_text()` (*GLXCurses.Entry* method), 198  
`get_placeholder_text()` (*GLXCurses.Entry.Entry* method), 73  
`get_position()` (*GLXCurses.Editable* method), 188  
`get_position()` (*GLXCurses.Editable.Editable* method), 63  
`get_progress_fraction()` (*GLXCurses.Entry* method), 200  
`get_progress_fraction()` (*GLXCurses.Entry.Entry* method), 75  
`get_progress_pulse_step()` (*GLXCurses.Entry* method), 200  
`get_progress_pulse_step()` (*GLXCurses.Entry.Entry* method), 75  
`get_range_rect()` (*GLXCurses.Range* method), 208  
`get_range_rect()` (*GLXCurses.Range.Range* method), 105  
`get_resize_mode()` (*GLXCurses.Container* method), 140  
`get_resize_mode()` (*GLXCurses.Container.Container* method), 54  
`get_response_for_widget()` (*GLXCurses.Dialog* method), 164  
`get_response_for_widget()` (*GLXCurses.Dialog.Dialog* method), 59  
`get_restrict_to_fill_level()` (*GLXCurses.Range* method), 205  
`get_restrict_to_fill_level()` (*GLXCurses.Range.Range* method), 102  
`get_round_digits()` (*GLXCurses.Range* method), 207  
`get_round_digits()` (*GLXCurses.Range.Range* method), 104  
`get_selectable()` (*GLXCurses.Label* method), 180  
`get_selectable()` (*GLXCurses.Label.Label* method), 92  
`get_selection_bounds()` (*GLXCurses.Editable* method), 185  
`get_selection_bounds()` (*GLXCurses.Editable.Editable* method), 61  
`get_shadow_type()` (*GLXCurses.Frame* method), 169  
`get_shadow_type()` (*GLXCurses.Frame.Frame* method), 86  
`get_show_fill_level()` (*GLXCurses.Range* method), 205  
`get_show_fill_level()` (*GLXCurses.Range.Range* method), 102



`get_single_line_mode()` (*GLXCurses.Label method*), 181  
`get_single_line_mode()` (*GLXCurses.Label.Label method*), 94  
`get_size()` (*GLXCurses.TextTagTable method*), 146  
`get_size()` (*GLXCurses.TextTagTable.TextTagTable method*), 110  
`get_slider_range()` (*GLXCurses.Range method*), 208  
`get_slider_range()` (*GLXCurses.Range.Range method*), 105  
`get_slider_size_fixed()` (*GLXCurses.Range method*), 208  
`get_slider_size_fixed()` (*GLXCurses.Range.Range method*), 105  
`get_states()` (*GLXCurses.Entry method*), 202  
`get_states()` (*GLXCurses.Entry.Entry method*), 77  
`get_step_increment()` (*GLXCurses.Adjustment method*), 161  
`get_step_increment()` (*GLXCurses.Adjustment.Adjustment method*), 39  
`get_tabs()` (*GLXCurses.Entry method*), 201  
`get_tabs()` (*GLXCurses.Entry.Entry method*), 76  
`get_tempo()` (*GLXCurses.Buzzer.Buzzer method*), 49  
`get_tempo_to_hertz()` (*GLXCurses.Buzzer.Buzzer method*), 50  
`get_tempo_to_ms()` (*GLXCurses.Buzzer.Buzzer method*), 49  
`get_text()` (*GLXCurses.Entry method*), 195  
`get_text()` (*GLXCurses.Entry.Entry method*), 70  
`get_text()` (*GLXCurses.EntryBuffer method*), 183  
`get_text()` (*GLXCurses.EntryBuffer.EntryBuffer method*), 78  
`get_text()` (*GLXCurses.Label method*), 180  
`get_text()` (*GLXCurses.Label.Label method*), 92  
`get_text_length()` (*GLXCurses.Entry method*), 195  
`get_text_length()` (*GLXCurses.Entry.Entry method*), 70  
`get_tooltip()` (*GLXCurses.libs.Spot.Spot method*), 24  
`get_toplevel()` (*GLXCurses.Widget method*), 138  
`get_toplevel()` (*GLXCurses.Widget.Widget method*), 119  
`get_triolet()` (*GLXCurses.Buzzer.Buzzer method*), 50  
`get_triple_croche()` (*GLXCurses.Buzzer.Buzzer method*), 50  
`get_upper()` (*GLXCurses.Adjustment method*), 161  
`get_upper()` (*GLXCurses.Adjustment.Adjustment method*), 40  
`get_upper_stepper_sensitivity()` (*GLXCurses.Range method*), 208  
`get_upper_stepper_sensitivity()` (*GLXCurses.Range.Range method*), 104  
`get_use_underline()` (*GLXCurses.Label method*), 181  
`get_use_underline()` (*GLXCurses.Label.Label method*), 93  
`get_value()` (*GLXCurses.Adjustment method*), 159  
`get_value()` (*GLXCurses.Adjustment.Adjustment method*), 38  
`get_value()` (*GLXCurses.Range method*), 206  
`get_value()` (*GLXCurses.Range.Range method*), 103  
`get_visibility()` (*GLXCurses.Entry method*), 199  
`get_visibility()` (*GLXCurses.Entry.Entry method*), 74  
`get_widget_by_id()` (*GLXCurses.Application method*), 165  
`get_widget_by_id()` (*GLXCurses.Application.Application method*), 42  
`get_widget_for_response()` (*GLXCurses.Dialog method*), 165  
`get_widget_for_response()` (*GLXCurses.Dialog.Dialog method*), 59  
`get_width_chars()` (*GLXCurses.Entry method*), 196  
`get_width_chars()` (*GLXCurses.Entry.Entry method*), 71  
`get_width_chars()` (*GLXCurses.Label method*), 181  
`get_width_chars()` (*GLXCurses.Label.Label method*), 94  
`get_window_by_id()` (*GLXCurses.Application method*), 167  
`get_window_by_id()` (*GLXCurses.Application.Application method*), 44  
`get_window_type()` (*GLXCurses.Window method*), 157  
`get_window_type()` (*GLXCurses.Window.Window method*), 125  
`glxc_type()` (*in module GLXCurses.libs.Utils*), 30  
*GLXCurses (module)*, 125  
*GLXCurses.Actionable (module)*, 35  
*GLXCurses.Adjustment (module)*, 36  
*GLXCurses.Aera (module)*, 41  
*GLXCurses.Application (module)*, 42  
*GLXCurses.Bin (module)*, 44  
*GLXCurses.Bindings (module)*, 45  
*GLXCurses.Box (module)*, 45  
*GLXCurses.Button (module)*, 48  
*GLXCurses.Buzzer (module)*, 49  
*GLXCurses.CheckButton (module)*, 50  
*GLXCurses.Clipboards (module)*, 51  
*GLXCurses.Constants (module)*, 52

- GLXCurses.Container (module), 52  
 GLXCurses.Dialog (module), 57  
 GLXCurses.Editable (module), 60  
 GLXCurses.Entry (module), 64  
 GLXCurses.EntryBuffer (module), 78  
 GLXCurses.EntryCompletion (module), 80  
 GLXCurses.EventList (module), 82  
 GLXCurses.FileChooser (module), 82  
 GLXCurses.FileChooserMenu (module), 83  
 GLXCurses.Frame (module), 84  
 GLXCurses.HBox (module), 86  
 GLXCurses.HSeparator (module), 86  
 GLXCurses.Image (module), 87  
 GLXCurses.Label (module), 88  
 GLXCurses.libs (module), 35  
 GLXCurses.libs.ApplicationHandlers (module), 15  
 GLXCurses.libs.ChildElement (module), 15  
 GLXCurses.libs.ChildProperty (module), 16  
 GLXCurses.libs.Colorable (module), 17  
 GLXCurses.libs.Colors (module), 17  
 GLXCurses.libs.Dividable (module), 19  
 GLXCurses.libs.File (module), 19  
 GLXCurses.libs.FileChooserFunctions (module), 20  
 GLXCurses.libs.Group (module), 21  
 GLXCurses.libs.GroupElement (module), 22  
 GLXCurses.libs.Groups (module), 22  
 GLXCurses.libs.handlers (module), 15  
 GLXCurses.libs.handlers.application (module), 14  
 GLXCurses.libs.handlers.button (module), 14  
 GLXCurses.libs.handlers.container (module), 14  
 GLXCurses.libs.handlers.editable (module), 14  
 GLXCurses.libs.handlers.filechooser (module), 14  
 GLXCurses.libs.handlers.label (module), 15  
 GLXCurses.libs.handlers.statusbar (module), 15  
 GLXCurses.libs.handlers.textview (module), 15  
 GLXCurses.libs.handlers.widget (module), 15  
 GLXCurses.libs.handlers.window (module), 15  
 GLXCurses.libs.ImageConvert (module), 23  
 GLXCurses.libs.Movable (module), 24  
 GLXCurses.libs.Spot (module), 24  
 GLXCurses.libs.TextAttributes (module), 28  
 GLXCurses.libs.TextFonts (module), 29  
 GLXCurses.libs.TextUtils (module), 29  
 GLXCurses.libs.TTY (module), 25  
 GLXCurses.libs.Utils (module), 30  
 GLXCurses.libs.XDGBaseDirectory (module), 33  
 GLXCurses.MainLoop (module), 94  
 GLXCurses.Menu (module), 95  
 GLXCurses.MenuBar (module), 95  
 GLXCurses.MenuItem (module), 95  
 GLXCurses.MenuShell (module), 96  
 GLXCurses.MessageBar (module), 96  
 GLXCurses.Misc (module), 98  
 GLXCurses.Object (module), 99  
 GLXCurses.ProgressBar (module), 100  
 GLXCurses.RadioButton (module), 100  
 GLXCurses.Range (module), 100  
 GLXCurses.Scrollable (module), 105  
 GLXCurses.StatusBar (module), 105  
 GLXCurses.Style (module), 107  
 GLXCurses.TextBuffer (module), 108  
 GLXCurses.TextTag (module), 108  
 GLXCurses.TextTagTable (module), 110  
 GLXCurses.TextView (module), 110  
 GLXCurses.ToolBar (module), 112  
 GLXCurses.VBox (module), 113  
 GLXCurses.VSeparator (module), 113  
 GLXCurses.VuMeter (module), 113  
 GLXCurses.Widget (module), 113  
 GLXCurses.Window (module), 120  
 glxcurses\_support (GLXCurses.MainLoop.MainLoop attribute), 94  
 grab\_focus\_without\_selecting() (GLXCurses.Entry method), 202  
 grab\_focus\_without\_selecting() (GLXCurses.Entry.Entry method), 77  
 gravity (GLXCurses.Window attribute), 153  
 gravity (GLXCurses.Window.Window attribute), 121  
 Group (class in GLXCurses.libs.Group), 21  
 group (GLXCurses.libs.Groups.Groups attribute), 22  
 GroupElement (class in GLXCurses.libs.GroupElement), 22  
 Groups (class in GLXCurses.libs.Groups), 22  
 groups (GLXCurses.libs.Groups.Groups attribute), 22  
**H**  
 halfdelay (GLXCurses.libs.TTY.Screen attribute), 26  
 halfdelay (GLXCurses.Screen attribute), 127  
 halign (GLXCurses.Widget attribute), 133  
 halign (GLXCurses.Widget.Widget attribute), 115  
 handle\_event() (GLXCurses.MainLoop.MainLoop method), 95  
 Handlers (class in GLXCurses.libs.ApplicationHandlers), 15

HandlersApplication (class in *Curses.libs.handlers.application*), 14  
 HandlersButton (class in *Curses.libs.handlers.button*), 14  
 HandlersContainer (class in *Curses.libs.handlers.container*), 14  
 HandlersEditable (class in *Curses.libs.handlers.editable*), 14  
 HandlersFileChooser (class in *Curses.libs.handlers.filechooser*), 14  
 HandlersLabel (class in *Curses.libs.handlers.label*), 15  
 HandlersStatusbar (class in *Curses.libs.handlers.statusbar*), 15  
 HandlersTextView (class in *Curses.libs.handlers.textview*), 15  
 HandlersWidget (class in *Curses.libs.handlers.widget*), 15  
 HandlersWindow (class in *Curses.libs.handlers.window*), 15  
 has\_default (*GLXCurses.libs.Spot.Spot* attribute), 24  
 has\_default (*GLXCurses.Widget* attribute), 134  
 has\_default (*GLXCurses.Widget.Widget* attribute), 115  
 has\_focus (*GLXCurses.libs.Spot.Spot* attribute), 24  
 has\_focus (*GLXCurses.Widget* attribute), 134  
 has\_focus (*GLXCurses.Widget.Widget* attribute), 115  
 has\_frame (*GLXCurses.Entry* attribute), 190  
 has\_frame (*GLXCurses.Entry.Entry* attribute), 66  
 has\_prelight (*GLXCurses.libs.Spot.Spot* attribute), 24  
 has\_prelight (*GLXCurses.Widget* attribute), 134  
 has\_prelight (*GLXCurses.Widget.Widget* attribute), 115  
 has\_resize\_grip (*GLXCurses.Window* attribute), 153  
 has\_resize\_grip (*GLXCurses.Window.Window* attribute), 122  
 has\_selection (*GLXCurses.TextBuffer* attribute), 146  
 has\_selection (*GLXCurses.TextBuffer.TextBuffer* attribute), 108  
 has\_tooltip (*GLXCurses.Widget* attribute), 134  
 has\_tooltip (*GLXCurses.Widget.Widget* attribute), 115  
 has\_toplevel\_focus (*GLXCurses.Window* attribute), 154  
 has\_toplevel\_focus (*GLXCurses.Window.Window* attribute), 122  
 has\_window (*GLXCurses.Widget* attribute), 138  
 has\_window (*GLXCurses.Widget.Widget* attribute), 120  
 HBox (class in *GLXCurses*), 151  
 HBox (class in *GLXCurses.HBox*), 86  
 GLX- height (*GLXCurses.Aera.Area* attribute), 41  
 GLX- height (*GLXCurses.libs.TextUtils.TextUtils* attribute), 29  
 height\_max (*GLXCurses.Image* attribute), 211  
 GLX- height\_max (*GLXCurses.Image.Image* attribute), 87  
 GLX- height\_max (*GLXCurses.ImageConvert* attribute), 212  
 height\_max (*GLXCurses.libs.ImageConvert.ImageConvert* attribute), 23  
 GLX- height\_original (*GLXCurses.Image* attribute), 211  
 GLX- height\_original (*GLXCurses.Image.Image* attribute), 87  
 GLX- height\_original (*GLXCurses.ImageConvert* attribute), 212  
 GLX- height\_original (*GLXCurses.libs.ImageConvert.ImageConvert* attribute), 23  
 height\_request (*GLXCurses.Widget* attribute), 134  
 GLX- height\_request (*GLXCurses.Widget.Widget* attribute), 115  
 hex\_rgb\_to\_curses () (*GLXCurses.Colors* method), 131  
 hex\_rgb\_to\_curses () (*GLXCurses.libs.Colors.Colors* method), 18  
 hexpand (*GLXCurses.Widget* attribute), 134  
 hexpand (*GLXCurses.Widget.Widget* attribute), 115  
 hexpand\_set (*GLXCurses.Widget* attribute), 134  
 hexpand\_set (*GLXCurses.Widget.Widget* attribute), 115  
 hide () (*GLXCurses.Widget* method), 138  
 hide () (*GLXCurses.Widget.Widget* method), 119  
 hide\_titlebar\_when\_maximized (*GLXCurses.Window* attribute), 154  
 hide\_titlebar\_when\_maximized (*GLXCurses.Window.Window* attribute), 122  
 history\_box\_num\_cols (*GLXCurses.FileChooserMenu.FileChooserMenu* attribute), 83  
 history\_dir\_list (*GLXCurses.FileChooserMenu.FileChooserMenu* attribute), 83  
 homogeneous (*GLXCurses.Box* attribute), 148  
 homogeneous (*GLXCurses.Box.Box* attribute), 45  
 HSeparator (class in *GLXCurses*), 182  
 HSeparator (class in *GLXCurses.HSeparator*), 86  
 hsp\_debug (*GLXCurses.Image* attribute), 211  
 hsp\_debug (*GLXCurses.Image.Image* attribute), 87  
 hsp\_debug (*GLXCurses.ImageConvert* attribute), 212  
 hsp\_debug (*GLXCurses.libs.ImageConvert.ImageConvert* attribute), 23  
 |  
 icon (*GLXCurses.Window* attribute), 154  
 icon (*GLXCurses.Window.Window* attribute), 122

`icon_name` (*GLXCurses.Window* attribute), 154  
`icon_name` (*GLXCurses.Window.Window* attribute), 122  
`id` (*GLXCurses.libs.ChildElement.ChildElement* attribute), 16  
`id` (*GLXCurses.Object* attribute), 132  
`id` (*GLXCurses.Object.Object* attribute), 99  
`im_context_filter_keypress()` (*GLXCurses.Entry* method), 200  
`im_context_filter_keypress()` (*GLXCurses.Entry.Entry* method), 75  
`im_module` (*GLXCurses.Entry* attribute), 190  
`im_module` (*GLXCurses.Entry.Entry* attribute), 66  
`Image` (class in *GLXCurses*), 211  
`Image` (class in *GLXCurses.Image*), 87  
`image_object` (*GLXCurses.Image* attribute), 211  
`image_object` (*GLXCurses.Image.Image* attribute), 87  
`ImageConvert` (class in *GLXCurses*), 212  
`ImageConvert` (class in *GLXCurses.libs.ImageConvert*), 23  
`in_destruction` (*GLXCurses.Widget* attribute), 137  
`in_destruction` (*GLXCurses.Widget.Widget* attribute), 118  
`indent` (*GLXCurses.TextView* attribute), 147  
`indent` (*GLXCurses.TextView.TextView* attribute), 111  
`info_label` (*GLXCurses.MenuBar* attribute), 170  
`info_label` (*GLXCurses.MenuBar.MenuBar* attribute), 95  
`init_button_positions()` (*GLXCurses.ToolBar* method), 174  
`init_button_positions()` (*GLXCurses.ToolBar.ToolBar* method), 112  
`inline_completion` (*GLXCurses.EntryCompletion* attribute), 202  
`inline_completion` (*GLXCurses.EntryCompletion.EntryCompletion* attribute), 81  
`inline_selection` (*GLXCurses.EntryCompletion* attribute), 203  
`inline_selection` (*GLXCurses.EntryCompletion.EntryCompletion* attribute), 81  
`inner_border` (*GLXCurses.Entry* attribute), 191  
`inner_border` (*GLXCurses.Entry.Entry* attribute), 66  
`input_hints` (*GLXCurses.Entry* attribute), 191  
`input_hints` (*GLXCurses.Entry.Entry* attribute), 66  
`input_hints` (*GLXCurses.TextView* attribute), 147  
`input_hints` (*GLXCurses.TextView.TextView* attribute), 111  
`input_purpose` (*GLXCurses.Entry* attribute), 191  
`input_purpose` (*GLXCurses.Entry.Entry* attribute), 66  
`input_purpose` (*GLXCurses.TextView* attribute), 147  
`input_purpose` (*GLXCurses.TextView.TextView* attribute), 111  
`insert_character()` (*GLXCurses.Aera.Area* method), 42  
`insert_string()` (*GLXCurses.Aera.Area* method), 42  
`insert_text()` (*GLXCurses.Editable* method), 186  
`insert_text()` (*GLXCurses.Editable.Editable* method), 61  
`insert_text()` (*GLXCurses.EntryBuffer* method), 184  
`insert_text()` (*GLXCurses.EntryBuffer.EntryBuffer* method), 79  
`instance` (*GLXCurses.Application* attribute), 166  
`instance` (*GLXCurses.Application.Application* attribute), 43  
`instance` (*GLXCurses.libs.TTY.Screen* attribute), 26  
`instance` (*GLXCurses.MainLoop.MainLoop* attribute), 94  
`instance` (*GLXCurses.Screen* attribute), 128  
`int` (built-in variable), 213  
`interface` (*GLXCurses.Button.Button* attribute), 49  
`interface` (*GLXCurses.CheckButton* attribute), 158  
`interface` (*GLXCurses.CheckButton.CheckButton* attribute), 51  
`interface` (*GLXCurses.RadioButton* attribute), 157  
`interface` (*GLXCurses.RadioButton.RadioButton* attribute), 100  
`interface_normal` (*GLXCurses.Button.Button* attribute), 48  
`interface_selected` (*GLXCurses.Button.Button* attribute), 48  
`intrflush` (*GLXCurses.libs.TTY.Screen* attribute), 26  
`intrflush` (*GLXCurses.Screen* attribute), 127  
`inverted` (*GLXCurses.Range* attribute), 204  
`inverted` (*GLXCurses.Range.Range* attribute), 100  
`invisible_char` (*GLXCurses.Entry* attribute), 191  
`invisible_char` (*GLXCurses.Entry.Entry* attribute), 67  
`invisible_char_set` (*GLXCurses.Entry* attribute), 192  
`invisible_char_set` (*GLXCurses.Entry.Entry* attribute), 67  
`is_accel` (*GLXCurses.MenuItem* attribute), 171  
`is_accel` (*GLXCurses.MenuItem.MenuItem* attribute), 96  
`is_active` (*GLXCurses.Window* attribute), 154  
`is_active` (*GLXCurses.Window.Window* attribute), 122  
`is_binary()` (*GLXCurses.libs.File.File* method), 19  
`is_focus` (*GLXCurses.Widget* attribute), 134  
`is_focus` (*GLXCurses.Widget.Widget* attribute), 116  
`is_group()` (*GLXCurses.libs.Groups.Groups* method), 22



- is\_maximized (*GLXCurses.Window* attribute), 154  
 is\_maximized (*GLXCurses.Window.Window* attribute), 122  
 is\_member () (*GLXCurses.libs.Group.Group* method), 22  
 is\_resized (*GLXCurses.Image* attribute), 211  
 is\_resized (*GLXCurses.Image.Image* attribute), 87  
 is\_resized (*GLXCurses.ImageConvert* attribute), 212  
 is\_resized (*GLXCurses.libs.ImageConvert.ImageConvert* attribute), 23  
 is\_text () (*GLXCurses.libs.File.File* method), 19  
 is\_valid\_id () (in module *GLXCurses.libs.Utils*), 33  
 item\_it\_can\_be\_display (*GLXCurses.FileChooser.FileSelect* attribute), 82  
 item\_it\_can\_be\_display (*GLXCurses.FileSelect* attribute), 210  
 item\_scroll\_pos (*GLXCurses.FileChooser.FileSelect* attribute), 82  
 item\_scroll\_pos (*GLXCurses.FileSelect* attribute), 210  
 itu\_recommendation (*GLXCurses.Colors* attribute), 130  
 itu\_recommendation (*GLXCurses.libs.Colors.Colors* attribute), 17
- ## J
- Justification (built-in variable), 214  
 justification (*GLXCurses.TextView* attribute), 147  
 justification (*GLXCurses.TextView.TextView* attribute), 111  
 justify (*GLXCurses.libs.Movable.Movable* attribute), 24
- ## K
- keypad (*GLXCurses.libs.TTY.Screen* attribute), 26  
 keypad (*GLXCurses.Screen* attribute), 127
- ## L
- Label (class in *GLXCurses*), 175  
 Label (class in *GLXCurses.Label*), 88  
 label (*GLXCurses.FileChooserMenu.FileChooserMenu* attribute), 83  
 label (*GLXCurses.Frame* attribute), 167  
 label (*GLXCurses.Frame.Frame* attribute), 84  
 label (*GLXCurses.Label* attribute), 176  
 label (*GLXCurses.Label.Label* attribute), 88  
 label (*GLXCurses.libs.TextAttributes.TextAttributes* attribute), 28  
 label (*GLXCurses.MenuItem* attribute), 170  
 label (*GLXCurses.MenuItem.MenuItem* attribute), 96  
 label\_widget (*GLXCurses.Frame* attribute), 168  
 label\_widget (*GLXCurses.Frame.Frame* attribute), 84  
 label\_xalign (*GLXCurses.Frame* attribute), 168  
 label\_xalign (*GLXCurses.Frame.Frame* attribute), 84  
 label\_yalign (*GLXCurses.Frame* attribute), 168  
 label\_yalign (*GLXCurses.Frame.Frame* attribute), 84  
 labels (*GLXCurses.ToolBar* attribute), 174  
 labels (*GLXCurses.ToolBar.ToolBar* attribute), 112  
 layout\_index\_to\_text\_index () (*GLXCurses.Entry* method), 198  
 layout\_index\_to\_text\_index () (*GLXCurses.Entry.Entry* method), 74  
 left\_margin (*GLXCurses.TextView* attribute), 147  
 left\_margin (*GLXCurses.TextView.TextView* attribute), 111  
 length (*GLXCurses.EntryBuffer* attribute), 182  
 length (*GLXCurses.EntryBuffer.EntryBuffer* attribute), 78  
 lines (*GLXCurses.Label* attribute), 176  
 lines (*GLXCurses.Label.Label* attribute), 88  
 lines (*GLXCurses.libs.TextUtils.TextUtils* attribute), 30  
 list (built-in variable), 213  
 load\_image () (*GLXCurses.Image* method), 211  
 load\_image () (*GLXCurses.Image.Image* method), 88  
 load\_image () (*GLXCurses.ImageConvert* method), 212  
 load\_image () (*GLXCurses.libs.ImageConvert.ImageConvert* method), 23  
 long (built-in variable), 213  
 lookup () (*GLXCurses.TextTagTable* method), 146  
 lookup () (*GLXCurses.TextTagTable.TextTagTable* method), 110  
 lower (*GLXCurses.Adjustment* attribute), 158  
 lower (*GLXCurses.Adjustment.Adjustment* attribute), 36  
 lower\_stepper\_sensitivity (*GLXCurses.Range* attribute), 204  
 lower\_stepper\_sensitivity (*GLXCurses.Range.Range* attribute), 101  
 lowlevel\_getch () (*GLXCurses.libs.TTY.Screen* method), 27  
 lowlevel\_getch () (*GLXCurses.Screen* method), 129
- ## M
- MainLoop (class in *GLXCurses.MainLoop*), 94  
 map (*GLXCurses.Widget* attribute), 138  
 map (*GLXCurses.Widget.Widget* attribute), 119  
 margin (*GLXCurses.Widget* attribute), 134  
 margin (*GLXCurses.Widget.Widget* attribute), 116  
 margin\_bottom (*GLXCurses.Widget* attribute), 135

`margin_bottom` (*GLXCurses.Widget.Widget attribute*), 116  
`margin_end` (*GLXCurses.Widget attribute*), 135  
`margin_end` (*GLXCurses.Widget.Widget attribute*), 116  
`margin_start` (*GLXCurses.Widget attribute*), 135  
`margin_start` (*GLXCurses.Widget.Widget attribute*), 116  
`margin_top` (*GLXCurses.Widget attribute*), 135  
`margin_top` (*GLXCurses.Widget.Widget attribute*), 116  
`markdown_is_used` (*GLXCurses.libs.TextAttributes.TextAttributes attribute*), 28  
`max_length` (*GLXCurses.Entry attribute*), 192  
`max_length` (*GLXCurses.Entry.Entry attribute*), 67  
`max_length` (*GLXCurses.EntryBuffer attribute*), 182  
`max_length` (*GLXCurses.EntryBuffer.EntryBuffer attribute*), 78  
`max_width_chars` (*GLXCurses.Entry attribute*), 192  
`max_width_chars` (*GLXCurses.Entry.Entry attribute*), 67  
`max_width_chars` (*GLXCurses.Label attribute*), 176  
`max_width_chars` (*GLXCurses.Label.Label attribute*), 88  
`members` (*GLXCurses.libs.Group.Group attribute*), 21  
`Menu` (*class in GLXCurses*), 170  
`Menu` (*class in GLXCurses.Menu*), 95  
`MenuBar` (*class in GLXCurses*), 170  
`MenuBar` (*class in GLXCurses.MenuBar*), 95  
`menubar` (*GLXCurses.Application attribute*), 166  
`menubar` (*GLXCurses.Application.Application attribute*), 43  
`MenuItem` (*class in GLXCurses*), 170  
`MenuItem` (*class in GLXCurses.MenuItem*), 95  
`MenuShell` (*class in GLXCurses*), 169  
`MenuShell` (*class in GLXCurses.MenuShell*), 96  
`merge_dicts()` (*in module GLXCurses.libs.Utils*), 33  
`MessageBar` (*class in GLXCurses*), 172  
`MessageBar` (*class in GLXCurses.MessageBar*), 96  
`messagebar` (*GLXCurses.Application attribute*), 166  
`messagebar` (*GLXCurses.Application.Application attribute*), 43  
`meta` (*GLXCurses.libs.TTY.Screen attribute*), 26  
`meta` (*GLXCurses.Screen attribute*), 128  
`minimum_increment` (*GLXCurses.Adjustment attribute*), 158  
`minimum_increment` (*GLXCurses.Adjustment.Adjustment attribute*), 37  
`minimum_key_length` (*GLXCurses.EntryCompletion attribute*), 203  
`minimum_key_length` (*GLXCurses.EntryCompletion.EntryCompletion attribute*), 81  
`Misc` (*class in GLXCurses*), 174  
`Misc` (*class in GLXCurses.Misc*), 98  
`mnemonic_char` (*GLXCurses.libs.TextAttributes.TextAttributes attribute*), 29  
`mnemonic_is_used` (*GLXCurses.libs.TextAttributes.TextAttributes attribute*), 29  
`mnemonic_keyval` (*GLXCurses.Label attribute*), 176  
`mnemonic_keyval` (*GLXCurses.Label.Label attribute*), 89  
`mnemonic_use_underline` (*GLXCurses.libs.TextAttributes.TextAttributes attribute*), 29  
`mnemonic_widget` (*GLXCurses.Label attribute*), 176  
`mnemonic_widget` (*GLXCurses.Label.Label attribute*), 89  
`mnemonics_visible` (*GLXCurses.Window attribute*), 154  
`mnemonics_visible` (*GLXCurses.Window.Window attribute*), 122  
`modal` (*GLXCurses.Window attribute*), 154  
`modal` (*GLXCurses.Window.Window attribute*), 123  
`model` (*GLXCurses.EntryCompletion attribute*), 203  
`model` (*GLXCurses.EntryCompletion.EntryCompletion attribute*), 81  
`model` (*GLXCurses.Range attribute*), 204  
`model` (*GLXCurses.Range.Range attribute*), 101  
`Movable` (*class in GLXCurses.libs.Movable*), 24  
`MovementStep` (*built-in variable*), 215  
`mtime_column_width` (*GLXCurses.FileChooser.FileSelect attribute*), 83  
`mtime_column_width` (*GLXCurses.FileSelect attribute*), 210

## N

`name` (*GLXCurses.libs.ChildElement.ChildElement attribute*), 16  
`name` (*GLXCurses.libs.File.File attribute*), 19  
`name` (*GLXCurses.Widget attribute*), 135  
`name` (*GLXCurses.Widget.Widget attribute*), 116  
`name_column_width` (*GLXCurses.FileChooser.FileSelect attribute*), 83  
`name_column_width` (*GLXCurses.FileSelect attribute*), 210  
`new()` (*GLXCurses.Adjustment method*), 159  
`new()` (*GLXCurses.Adjustment.Adjustment method*), 38  
`new()` (*GLXCurses.Box method*), 148  
`new()` (*GLXCurses.Box.Box method*), 46  
`new()` (*GLXCurses.Entry method*), 194  
`new()` (*GLXCurses.Entry.Entry method*), 69

new() (*GLXCurses.EntryBuffer* method), 183  
 new() (*GLXCurses.EntryBuffer.EntryBuffer* method), 78  
 new() (*GLXCurses.EntryCompletion* method), 203  
 new() (*GLXCurses.EntryCompletion.EntryCompletion* method), 82  
 new() (*GLXCurses.Frame* method), 168  
 new() (*GLXCurses.Frame.Frame* method), 84  
 new() (*GLXCurses.HBox* method), 151  
 new() (*GLXCurses.HBox.HBox* method), 86  
 new() (*GLXCurses.Label* method), 178  
 new() (*GLXCurses.Label.Label* method), 90  
 new() (*GLXCurses.libs.TextAttributes.TextAttributes* method), 29  
 new() (*GLXCurses.MessageBar* method), 173  
 new() (*GLXCurses.MessageBar.MessageBar* method), 97  
 new() (*GLXCurses.StatusBar* method), 171  
 new() (*GLXCurses.StatusBar.StatusBar* method), 106  
 new() (*GLXCurses.TextTagTable* method), 145  
 new() (*GLXCurses.TextTagTable.TextTagTable* method), 110  
 new() (*GLXCurses.VBox* method), 151  
 new() (*GLXCurses.VBox.VBox* method), 113  
 new() (*GLXCurses.Widget* method), 137  
 new() (*GLXCurses.Widget.Widget* method), 118  
 new\_id() (in module *GLXCurses.libs.Utils*), 32  
 new\_with\_buffer() (*GLXCurses.Entry* method), 194  
 new\_with\_buffer() (*GLXCurses.Entry.Entry* method), 69  
 new\_with\_buttons() (*GLXCurses.Dialog* method), 163  
 new\_with\_buttons() (*GLXCurses.Dialog.Dialog* method), 58  
 new\_with\_mnemonic() (*GLXCurses.Label* method), 180  
 new\_with\_mnemonic() (*GLXCurses.Label.Label* method), 92  
 no\_show\_all (*GLXCurses.Widget* attribute), 135  
 no\_show\_all (*GLXCurses.Widget.Widget* attribute), 117  
 nodelay (*GLXCurses.libs.TTY.Screen* attribute), 27  
 nodelay (*GLXCurses.Screen* attribute), 128  
 None (built-in variable), 213

## O

Object (class in *GLXCurses*), 131  
 Object (class in *GLXCurses.Object*), 99  
 Orientation (built-in variable), 215  
 override\_background\_color() (*GLXCurses.Widget* method), 138  
 override\_background\_color() (*GLXCurses.Widget.Widget* method), 119

override\_color() (*GLXCurses.Widget* method), 138  
 override\_color() (*GLXCurses.Widget.Widget* method), 119  
 overwrite (*GLXCurses.libs.File.File* attribute), 19  
 overwrite (*GLXCurses.TextView* attribute), 147  
 overwrite (*GLXCurses.TextView.TextView* attribute), 111  
 overwrite\_mode (*GLXCurses.Entry* attribute), 192  
 overwrite\_mode (*GLXCurses.Entry.Entry* attribute), 67

## P

pack\_end() (*GLXCurses.Box* method), 149  
 pack\_end() (*GLXCurses.Box.Box* method), 46  
 pack\_start() (*GLXCurses.Box* method), 149  
 pack\_start() (*GLXCurses.Box.Box* method), 46  
 pack\_type (*GLXCurses.libs.ChildProperty.ChildProperty* attribute), 16  
 PackType (built-in variable), 215  
 padding (*GLXCurses.libs.ChildProperty.ChildProperty* attribute), 16  
 page\_increment (*GLXCurses.Adjustment* attribute), 158  
 page\_increment (*GLXCurses.Adjustment.Adjustment* attribute), 37  
 page\_size (*GLXCurses.Adjustment* attribute), 158  
 page\_size (*GLXCurses.Adjustment.Adjustment* attribute), 37  
 parent (*GLXCurses.Widget* attribute), 135  
 parent (*GLXCurses.Widget.Widget* attribute), 117  
 parse() (*GLXCurses.libs.TextAttributes.TextAttributes* method), 29  
 parse\_markdown\_with\_mnemonic() (*GLXCurses.libs.TextAttributes.TextAttributes* method), 29  
 parse\_markdown\_with\_no\_mnemonic() (*GLXCurses.libs.TextAttributes.TextAttributes* method), 29  
 parse\_text() (*GLXCurses.libs.TextAttributes.TextAttributes* method), 29  
 paste\_clipboard() (*GLXCurses.Editable* method), 187  
 paste\_clipboard() (*GLXCurses.Editable.Editable* method), 62  
 path (*GLXCurses.libs.File.File* attribute), 19  
 pattern (*GLXCurses.Label* attribute), 176  
 pattern (*GLXCurses.Label.Label* attribute), 89  
 permit\_keyboard\_interruption (*GLXCurses.libs.Spot.Spot* attribute), 25  
 placeholder\_text (*GLXCurses.Entry* attribute), 192

- placeholder\_text (*GLXCurses.Entry.Entry* attribute), 67
- play\_sound() (in module *GLXCurses.Buzzer*), 49
- pop() (*GLXCurses.EventList.EventList* method), 82
- pop() (*GLXCurses.MessageBar* method), 173
- pop() (*GLXCurses.MessageBar.MessageBar* method), 97
- pop() (*GLXCurses.StatusBar* method), 172
- pop() (*GLXCurses.StatusBar.StatusBar* method), 106
- populate\_all (*GLXCurses.Entry* attribute), 192
- populate\_all (*GLXCurses.Entry.Entry* attribute), 68
- populate\_all (*GLXCurses.TextView* attribute), 147
- populate\_all (*GLXCurses.TextView.TextView* attribute), 111
- popup\_completion (*GLXCurses.EntryCompletion* attribute), 203
- popup\_completion (*GLXCurses.EntryCompletion.EntryCompletion* attribute), 81
- popup\_single\_match (*GLXCurses.EntryCompletion* attribute), 203
- popup\_single\_match (*GLXCurses.EntryCompletion.EntryCompletion* attribute), 81
- position (*GLXCurses.libs.ChildProperty.ChildProperty* attribute), 17
- position (*GLXCurses.libs.Group.Group* attribute), 21
- position (*GLXCurses.libs.Groups.Groups* attribute), 22
- position (*GLXCurses.Window* attribute), 156
- position (*GLXCurses.Window.Window* attribute), 124
- position\_type (*GLXCurses.libs.Movable.Movable* attribute), 24
- PositionType (built-in variable), 215
- preferred\_height (*GLXCurses.Widget* attribute), 132
- preferred\_height (*GLXCurses.Widget.Widget* attribute), 113
- preferred\_width (*GLXCurses.Widget* attribute), 132
- preferred\_width (*GLXCurses.Widget.Widget* attribute), 113
- prepare\_attributes() (*GLXCurses.libs.TextAttributes.TextAttributes* method), 29
- primary\_icon\_activatable (*GLXCurses.Entry* attribute), 193
- primary\_icon\_activatable (*GLXCurses.Entry.Entry* attribute), 68
- primary\_icon\_gicon (*GLXCurses.Entry* attribute), 193
- primary\_icon\_gicon (*GLXCurses.Entry.Entry* attribute), 68
- primary\_icon\_name (*GLXCurses.Entry* attribute), 193
- primary\_icon\_name (*GLXCurses.Entry.Entry* attribute), 68
- primary\_icon\_sensitive (*GLXCurses.Entry* attribute), 193
- primary\_icon\_sensitive (*GLXCurses.Entry.Entry* attribute), 68
- primary\_icon\_tooltip\_markdown (*GLXCurses.Entry* attribute), 193
- primary\_icon\_tooltip\_markdown (*GLXCurses.Entry.Entry* attribute), 69
- primary\_icon\_tooltip\_text (*GLXCurses.Entry* attribute), 193
- primary\_icon\_tooltip\_text (*GLXCurses.Entry.Entry* attribute), 69
- progress\_fraction (*GLXCurses.Entry* attribute), 194
- progress\_fraction (*GLXCurses.Entry.Entry* attribute), 69
- progress\_pulse() (*GLXCurses.Entry* method), 200
- progress\_pulse() (*GLXCurses.Entry.Entry* method), 75
- ProgressBar (class in *GLXCurses*), 182
- ProgressBar (class in *GLXCurses.ProgressBar*), 100
- propagate\_expose() (*GLXCurses.Container* method), 141
- propagate\_expose() (*GLXCurses.Container.Container* method), 55
- properties (*GLXCurses.libs.ChildElement.ChildElement* attribute), 16
- push() (*GLXCurses.MessageBar* method), 173
- push() (*GLXCurses.MessageBar.MessageBar* method), 97
- push() (*GLXCurses.StatusBar* method), 172
- push() (*GLXCurses.StatusBar.StatusBar* method), 106
- ## Q
- qiflush (*GLXCurses.libs.TTY.Screen* attribute), 27
- qiflush (*GLXCurses.Screen* attribute), 128
- query\_child\_packing() (*GLXCurses.Box* method), 150
- query\_child\_packing() (*GLXCurses.Box.Box* method), 47
- ## R
- RadioButton (class in *GLXCurses*), 157
- RadioButton (class in *GLXCurses.RadioButton*), 100
- Range (class in *GLXCurses*), 203
- Range (class in *GLXCurses.Range*), 100
- raw (*GLXCurses.libs.TTY.Screen* attribute), 27
- raw (*GLXCurses.Screen* attribute), 128
- realize (*GLXCurses.Widget* attribute), 138
- realize (*GLXCurses.Widget.Widget* attribute), 119



- receives\_default (*GLXCurses.Widget attribute*), 136
- receives\_default (*GLXCurses.Widget.Widget attribute*), 117
- refresh() (*GLXCurses.Application method*), 167
- refresh() (*GLXCurses.Application.Application method*), 44
- refresh() (*GLXCurses.libs.TTY.Screen method*), 28
- refresh() (*GLXCurses.Screen method*), 129
- refresh() (*GLXCurses.Widget method*), 138
- refresh() (*GLXCurses.Widget.Widget method*), 119
- register\_session (*GLXCurses.Application attribute*), 166
- register\_session (*GLXCurses.Application.Application attribute*), 43
- ReliefStyle (*built-in variable*), 216
- remove() (*GLXCurses.Container method*), 140
- remove() (*GLXCurses.Container.Container method*), 53
- remove() (*GLXCurses.libs.Group.Group method*), 22
- remove() (*GLXCurses.MessageBar method*), 173
- remove() (*GLXCurses.MessageBar.MessageBar method*), 97
- remove() (*GLXCurses.StatusBar method*), 172
- remove() (*GLXCurses.StatusBar.StatusBar method*), 106
- remove() (*GLXCurses.TextTagTable method*), 146
- remove() (*GLXCurses.TextTagTable.TextTagTable method*), 110
- remove\_accel\_group() (*GLXCurses.Menu static method*), 170
- remove\_accel\_group() (*GLXCurses.Menu.Menu static method*), 95
- remove\_accel\_group() (*GLXCurses.Window static method*), 156
- remove\_accel\_group() (*GLXCurses.Window.Window static method*), 124
- remove\_all() (*GLXCurses.MessageBar method*), 173
- remove\_all() (*GLXCurses.MessageBar.MessageBar method*), 97
- remove\_all() (*GLXCurses.StatusBar method*), 172
- remove\_all() (*GLXCurses.StatusBar.StatusBar method*), 107
- remove\_group() (*GLXCurses.libs.Groups.Groups method*), 23
- remove\_window() (*GLXCurses.Application method*), 167
- remove\_window() (*GLXCurses.Application.Application method*), 44
- reorder\_child() (*GLXCurses.Box.Box method*), 47
- reset\_im\_context() (*GLXCurses.Entry method*), 200
- reset\_im\_context() (*GLXCurses.Entry.Entry method*), 76
- reset\_screen() (*GLXCurses.libs.TTY.Screen method*), 28
- reset\_screen() (*GLXCurses.Screen method*), 129
- resizable (*GLXCurses.Window attribute*), 155
- resizable (*GLXCurses.Window.Window attribute*), 123
- resize\_mode (*GLXCurses.Container attribute*), 139
- resize\_mode (*GLXCurses.Container.Container attribute*), 53
- resize\_text() (*in module GLXCurses.libs.Utils*), 32
- resize\_text\_wrap\_char() (*in module GLXCurses.libs.Utils*), 30
- resized\_text (*GLXCurses.MenuItem attribute*), 171
- resized\_text (*GLXCurses.MenuItem.MenuItem attribute*), 96
- resized\_text\_short\_cut (*GLXCurses.MenuItem attribute*), 171
- resized\_text\_short\_cut (*GLXCurses.MenuItem.MenuItem attribute*), 96
- resource (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory attribute*), 34
- response() (*GLXCurses.Dialog method*), 163
- response() (*GLXCurses.Dialog.Dialog method*), 58
- restrict\_to\_fill\_level (*GLXCurses.Range attribute*), 204
- restrict\_to\_fill\_level (*GLXCurses.Range.Range attribute*), 101
- rgb\_hex\_to\_list\_int() (*GLXCurses.Colors method*), 130
- rgb\_hex\_to\_list\_int() (*GLXCurses.libs.Colors.Colors method*), 18
- rgb\_to\_ansi16() (*GLXCurses.Colors static method*), 130
- rgb\_to\_ansi16() (*GLXCurses.libs.Colors.Colors static method*), 18
- rgb\_to\_curses\_attributes() (*GLXCurses.Colors method*), 130
- rgb\_to\_curses\_attributes() (*GLXCurses.libs.Colors.Colors method*), 18
- right\_justified (*GLXCurses.MenuItem attribute*), 171
- right\_justified (*GLXCurses.MenuItem.MenuItem attribute*), 96
- right\_margin (*GLXCurses.TextView attribute*), 147
- right\_margin (*GLXCurses.TextView.TextView attribute*), 111
- role (*GLXCurses.Window attribute*), 155
- role (*GLXCurses.Window.Window attribute*), 123
- round\_down() (*in module GLXCurses.libs.Utils*), 31

- `round_half_down()` (in module *GLXCurses.libs.Utils*), 31  
`round_half_up()` (in module *GLXCurses.libs.Utils*), 31  
`round_up()` (in module *GLXCurses.libs.Utils*), 31  
`run()` (*GLXCurses.Dialog* method), 163  
`run()` (*GLXCurses.Dialog.Dialog* method), 58  
`running` (*GLXCurses.MainLoop.MainLoop* attribute), 94
- ## S
- `scan()` (*GLXCurses.libs.TextUtils.TextUtils* method), 30  
`Screen` (class in *GLXCurses*), 126  
`Screen` (class in *GLXCurses.libs.TTY*), 25  
`screen` (*GLXCurses.Window* attribute), 155  
`screen` (*GLXCurses.Window.Window* attribute), 123  
`screensaver_active` (*GLXCurses.Application* attribute), 166  
`screensaver_active` (*GLXCurses.Application.Application* attribute), 43  
`ScrollStep` (built-in variable), 216  
`select_region()` (*GLXCurses.Editable* method), 185  
`select_region()` (*GLXCurses.Editable.Editable* method), 60  
`select_region()` (*GLXCurses.Label* method), 180  
`select_region()` (*GLXCurses.Label.Label* method), 93  
`selectable` (*GLXCurses.Label* attribute), 177  
`selectable` (*GLXCurses.Label.Label* attribute), 89  
`selected_item_info_list` (*GLXCurses.FileChooser.FileSelect* attribute), 82  
`selected_item_info_list` (*GLXCurses.FileSelect* attribute), 210  
`selected_item_pos` (*GLXCurses.FileChooser.FileSelect* attribute), 82  
`selected_item_pos` (*GLXCurses.FileSelect* attribute), 210  
`selected_menu` (*GLXCurses.MenuBar* attribute), 170  
`selected_menu` (*GLXCurses.MenuBar.MenuBar* attribute), 95  
`selected_menu_item` (*GLXCurses.MenuBar* attribute), 170  
`selected_menu_item` (*GLXCurses.MenuBar.MenuBar* attribute), 95  
`selection_bound` (*GLXCurses.Label* attribute), 177  
`selection_bound` (*GLXCurses.Label.Label* attribute), 89  
`SelectionMode` (built-in variable), 217  
`sensitive` (*GLXCurses.Widget* attribute), 136  
`sensitive` (*GLXCurses.Widget.Widget* attribute), 117  
`set_action_name()` (*GLXCurses.Actionable* method), 209  
`set_action_name()` (*GLXCurses.Actionable.Actionable* method), 36  
`set_action_target_value()` (*GLXCurses.Actionable* method), 209  
`set_action_target_value()` (*GLXCurses.Actionable.Actionable* method), 36  
`set_activates_default()` (*GLXCurses.Entry* method), 196  
`set_activates_default()` (*GLXCurses.Entry.Entry* method), 72  
`set_adjustment()` (*GLXCurses.Range* method), 206  
`set_adjustment()` (*GLXCurses.Range.Range* method), 102  
`set_alignment()` (*GLXCurses.Entry* method), 198  
`set_alignment()` (*GLXCurses.Entry.Entry* method), 73  
`set_app_file_extensions()` (*GLXCurses.libs.FileChooserFunctions.FileChooserUtils* method), 21  
`set_attributes()` (*GLXCurses.Entry* method), 199  
`set_attributes()` (*GLXCurses.Entry.Entry* method), 74  
`set_attributes()` (*GLXCurses.Label* method), 178  
`set_attributes()` (*GLXCurses.Label.Label* method), 91  
`set_border_width()` (*GLXCurses.Container* method), 143  
`set_border_width()` (*GLXCurses.Container.Container* method), 57  
`set_buffer()` (*GLXCurses.Entry* method), 194  
`set_buffer()` (*GLXCurses.Entry.Entry* method), 70  
`set_can_store()` (*GLXCurses.Clipboard* method), 126  
`set_can_store()` (*GLXCurses.Clipboards.Clipboard* method), 51  
`set_center_widget()` (*GLXCurses.Box* method), 151  
`set_center_widget()` (*GLXCurses.Box.Box* method), 48  
`set_child_packing()` (*GLXCurses.Box* method), 150  
`set_child_packing()` (*GLXCurses.Box.Box* method), 48  
`set_completion()` (*GLXCurses.Entry* method), 199  
`set_completion()` (*GLXCurses.Entry.Entry* method), 74  
`set_cursor_hadjustment()` (*GLXCurses.Entry* method), 199  
`set_cursor_hadjustment()` (*GLXCurses.Entry.Entry* method), 75

set\_decorated() (*GLXCurses.Widget method*), 138  
 set\_decorated() (*GLXCurses.Widget.Widget method*), 119  
 set\_default() (*GLXCurses.Window method*), 157  
 set\_default() (*GLXCurses.Window.Window method*), 125  
 set\_default\_response() (*GLXCurses.Dialog method*), 164  
 set\_default\_response() (*GLXCurses.Dialog.Dialog method*), 59  
 set\_editable() (*GLXCurses.Editable method*), 188  
 set\_editable() (*GLXCurses.Editable.Editable method*), 63  
 set\_fill\_level() (*GLXCurses.Range method*), 205  
 set\_fill\_level() (*GLXCurses.Range.Range method*), 102  
 set\_flippable() (*GLXCurses.Range method*), 208  
 set\_flippable() (*GLXCurses.Range.Range method*), 105  
 set\_focus\_chain() (*GLXCurses.Container method*), 141  
 set\_focus\_chain() (*GLXCurses.Container.Container method*), 55  
 set\_focus\_child() (*GLXCurses.Container method*), 142  
 set\_focus\_child() (*GLXCurses.Container.Container method*), 55  
 set\_focus\_hadjustment() (*GLXCurses.Container method*), 142  
 set\_focus\_hadjustment() (*GLXCurses.Container.Container method*), 55  
 set\_focus\_vadjustment() (*GLXCurses.Container method*), 142  
 set\_focus\_vadjustment() (*GLXCurses.Container.Container method*), 55  
 set\_has\_frame() (*GLXCurses.Entry method*), 197  
 set\_has\_frame() (*GLXCurses.Entry.Entry method*), 72  
 set\_icon\_activatable() (*GLXCurses.Entry method*), 201  
 set\_icon\_activatable() (*GLXCurses.Entry.Entry method*), 76  
 set\_icon\_drag\_source() (*GLXCurses.Entry method*), 202  
 set\_icon\_drag\_source() (*GLXCurses.Entry.Entry method*), 77  
 set\_icon\_from\_gicon() (*GLXCurses.Entry method*), 201  
 set\_icon\_from\_gicon() (*GLXCurses.Entry.Entry method*), 76  
 set\_icon\_from\_icon\_name() (*GLXCurses.Entry method*), 201  
 set\_icon\_from\_icon\_name() (*GLXCurses.Entry.Entry method*), 76  
 set\_icon\_from\_pixbuf() (*GLXCurses.Entry method*), 201  
 set\_icon\_from\_pixbuf() (*GLXCurses.Entry.Entry method*), 76  
 set\_icon\_from\_stock() (*GLXCurses.Entry method*), 201  
 set\_icon\_from\_stock() (*GLXCurses.Entry.Entry method*), 76  
 set\_icon\_sensitive() (*GLXCurses.Entry method*), 201  
 set\_icon\_sensitive() (*GLXCurses.Entry.Entry method*), 77  
 set\_icon\_tooltip\_markup() (*GLXCurses.Entry method*), 202  
 set\_icon\_tooltip\_markup() (*GLXCurses.Entry.Entry method*), 77  
 set\_icon\_tooltip\_text() (*GLXCurses.Entry method*), 201  
 set\_icon\_tooltip\_text() (*GLXCurses.Entry.Entry method*), 77  
 set\_increments() (*GLXCurses.Range method*), 206  
 set\_increments() (*GLXCurses.Range.Range method*), 103  
 set\_inner\_border() (*GLXCurses.Entry method*), 197  
 set\_inner\_border() (*GLXCurses.Entry.Entry method*), 72  
 set\_input\_hints() (*GLXCurses.Entry method*), 202  
 set\_input\_hints() (*GLXCurses.Entry.Entry method*), 77  
 set\_input\_purpose() (*GLXCurses.Entry method*), 202  
 set\_input\_purpose() (*GLXCurses.Entry.Entry method*), 77  
 set\_inverted() (*GLXCurses.Range method*), 206  
 set\_inverted() (*GLXCurses.Range.Range method*), 103  
 set\_invisible\_char() (*GLXCurses.Entry method*), 195  
 set\_invisible\_char() (*GLXCurses.Entry.Entry method*), 70  
 set\_justify() (*GLXCurses.Label method*), 179  
 set\_justify() (*GLXCurses.Label.Label method*), 91  
 set\_label() (*GLXCurses.Frame method*), 168  
 set\_label() (*GLXCurses.Frame.Frame method*), 84  
 set\_label\_align() (*GLXCurses.Frame method*), 168  
 set\_label\_align() (*GLXCurses.Frame.Frame method*), 85  
 set\_label\_text() (*GLXCurses.ToolBar method*), 174

`set_label_text()` (*GLXCurses.ToolBar.ToolBar method*), 112  
`set_label_widget()` (*GLXCurses.Frame method*), 168  
`set_label_widget()` (*GLXCurses.Frame.Frame method*), 85  
`set_line_wrap()` (*GLXCurses.Label method*), 179  
`set_line_wrap()` (*GLXCurses.Label.Label method*), 92  
`set_line_wrap_mode()` (*GLXCurses.Label method*), 179  
`set_line_wrap_mode()` (*GLXCurses.Label.Label method*), 92  
`set_lines()` (*GLXCurses.Label method*), 180  
`set_lines()` (*GLXCurses.Label.Label method*), 92  
`set_lower()` (*GLXCurses.Adjustment method*), 161  
`set_lower()` (*GLXCurses.Adjustment.Adjustment method*), 40  
`set_lower_stepper_sensitivity()` (*GLXCurses.Range method*), 207  
`set_lower_stepper_sensitivity()` (*GLXCurses.Range.Range method*), 104  
`set_markdown()` (*GLXCurses.Label method*), 178  
`set_markdown()` (*GLXCurses.Label.Label method*), 91  
`set_markdown_with_mnemonic()` (*GLXCurses.Label method*), 179  
`set_markdown_with_mnemonic()` (*GLXCurses.Label.Label method*), 91  
`set_max_length()` (*GLXCurses.Entry method*), 195  
`set_max_length()` (*GLXCurses.Entry.Entry method*), 71  
`set_max_length()` (*GLXCurses.EntryBuffer method*), 184  
`set_max_length()` (*GLXCurses.EntryBuffer.EntryBuffer method*), 79  
`set_max_width_chars()` (*GLXCurses.Entry method*), 197  
`set_max_width_chars()` (*GLXCurses.Entry.Entry method*), 73  
`set_max_width_chars()` (*GLXCurses.Label method*), 179  
`set_max_width_chars()` (*GLXCurses.Label.Label method*), 92  
`set_mnemonic_widget()` (*GLXCurses.Label method*), 181  
`set_mnemonic_widget()` (*GLXCurses.Label.Label method*), 93  
`set_overwrite_mode()` (*GLXCurses.Entry method*), 198  
`set_overwrite_mode()` (*GLXCurses.Entry.Entry method*), 73  
`set_page_increment()` (*GLXCurses.Adjustment method*), 161  
`set_page_increment()` (*GLXCurses.Adjustment.Adjustment method*), 40  
`set_page_size()` (*GLXCurses.Adjustment method*), 162  
`set_page_size()` (*GLXCurses.Adjustment.Adjustment method*), 40  
`set_pattern()` (*GLXCurses.Label method*), 179  
`set_pattern()` (*GLXCurses.Label.Label method*), 91  
`set_placeholder_text()` (*GLXCurses.Entry method*), 198  
`set_placeholder_text()` (*GLXCurses.Entry.Entry method*), 73  
`set_position()` (*GLXCurses.Editable method*), 187  
`set_position()` (*GLXCurses.Editable.Editable method*), 63  
`set_progress_fraction()` (*GLXCurses.Entry method*), 200  
`set_progress_fraction()` (*GLXCurses.Entry.Entry method*), 75  
`set_progress_pulse_step()` (*GLXCurses.Entry method*), 200  
`set_progress_pulse_step()` (*GLXCurses.Entry.Entry method*), 75  
`set_range()` (*GLXCurses.Range method*), 207  
`set_range()` (*GLXCurses.Range.Range method*), 103  
`set_reallocate_redraws()` (*GLXCurses.Container method*), 141  
`set_reallocate_redraws()` (*GLXCurses.Container.Container method*), 55  
`set_resize_mode()` (*GLXCurses.Container method*), 141  
`set_resize_mode()` (*GLXCurses.Container.Container method*), 54  
`set_resource()` (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory method*), 34  
`set_response_sensitive()` (*GLXCurses.Dialog method*), 164  
`set_response_sensitive()` (*GLXCurses.Dialog.Dialog method*), 59  
`set_restrict_to_fill_level()` (*GLXCurses.Range method*), 205  
`set_restrict_to_fill_level()` (*GLXCurses.Range.Range method*), 102  
`set_round_digits()` (*GLXCurses.Range method*), 207  
`set_round_digits()` (*GLXCurses.Range.Range method*), 104  
`set_selectable()` (*GLXCurses.Label method*), 181  
`set_selectable()` (*GLXCurses.Label.Label method*), 93



set\_shadow\_type() (*GLXCurses.Frame method*), 169  
 set\_shadow\_type() (*GLXCurses.Frame.Frame method*), 85  
 set\_show\_fill\_level() (*GLXCurses.Range method*), 206  
 set\_show\_fill\_level() (*GLXCurses.Range.Range method*), 102  
 set\_single\_line\_mode() (*GLXCurses.Label method*), 181  
 set\_single\_line\_mode() (*GLXCurses.Label.Label method*), 94  
 set\_slider\_size\_fixed() (*GLXCurses.Range method*), 209  
 set\_slider\_size\_fixed() (*GLXCurses.Range.Range method*), 105  
 set\_step\_increment() (*GLXCurses.Adjustment method*), 162  
 set\_step\_increment() (*GLXCurses.Adjustment.Adjustment method*), 41  
 set\_tabs() (*GLXCurses.Entry method*), 201  
 set\_tabs() (*GLXCurses.Entry.Entry method*), 76  
 set\_tempo() (*GLXCurses.Buzzer.Buzzer method*), 49  
 set\_text() (*GLXCurses.Clipboard method*), 126  
 set\_text() (*GLXCurses.Clipboards.Clipboard method*), 51  
 set\_text() (*GLXCurses.Entry method*), 194  
 set\_text() (*GLXCurses.Entry.Entry method*), 70  
 set\_text() (*GLXCurses.EntryBuffer method*), 183  
 set\_text() (*GLXCurses.EntryBuffer.EntryBuffer method*), 79  
 set\_text() (*GLXCurses.Label method*), 178  
 set\_text() (*GLXCurses.Label.Label method*), 90  
 set\_text\_with\_mnemonic() (*GLXCurses.Label method*), 181  
 set\_text\_with\_mnemonic() (*GLXCurses.Label.Label method*), 93  
 set\_tooltip() (*GLXCurses.libs.Spot.Spot method*), 25  
 set\_upper() (*GLXCurses.Adjustment method*), 162  
 set\_upper() (*GLXCurses.Adjustment.Adjustment method*), 41  
 set\_upper\_stepper\_sensitivity() (*GLXCurses.Range method*), 208  
 set\_upper\_stepper\_sensitivity() (*GLXCurses.Range.Range method*), 104  
 set\_use\_underline() (*GLXCurses.Label method*), 181  
 set\_use\_underline() (*GLXCurses.Label.Label method*), 93  
 set\_value() (*GLXCurses.Adjustment method*), 159  
 set\_value() (*GLXCurses.Adjustment.Adjustment method*), 38  
 set\_value() (*GLXCurses.Range method*), 206  
 set\_value() (*GLXCurses.Range.Range method*), 103  
 set\_visibility() (*GLXCurses.Entry method*), 195  
 set\_visibility() (*GLXCurses.Entry.Entry method*), 70  
 set\_width\_chars() (*GLXCurses.Entry method*), 197  
 set\_width\_chars() (*GLXCurses.Entry.Entry method*), 72  
 set\_width\_chars() (*GLXCurses.Label method*), 179  
 set\_width\_chars() (*GLXCurses.Label.Label method*), 92  
 set\_xalign() (*GLXCurses.Label method*), 179  
 set\_xalign() (*GLXCurses.Label.Label method*), 91  
 set\_yalign() (*GLXCurses.Label method*), 179  
 set\_yalign() (*GLXCurses.Label.Label method*), 91  
 shadow\_type (*GLXCurses.Frame attribute*), 168  
 shadow\_type (*GLXCurses.Frame.Frame attribute*), 84  
 ShadowType (*built-in variable*), 217  
 show() (*GLXCurses.Widget method*), 137  
 show() (*GLXCurses.Widget.Widget method*), 119  
 show\_all() (*GLXCurses.Widget method*), 138  
 show\_all() (*GLXCurses.Widget.Widget method*), 119  
 show\_fill\_level (*GLXCurses.Range attribute*), 205  
 show\_fill\_level (*GLXCurses.Range.Range attribute*), 101  
 show\_now() (*GLXCurses.Widget method*), 137  
 show\_now() (*GLXCurses.Widget.Widget method*), 119  
 show\_text (*GLXCurses.ProgressBar attribute*), 182  
 show\_text (*GLXCurses.ProgressBar.ProgressBar attribute*), 100  
 single\_line\_mode (*GLXCurses.Label attribute*), 177  
 single\_line\_mode (*GLXCurses.Label.Label attribute*), 89  
 Singleton (*class in GLXCurses.Application*), 42  
 Singleton (*class in GLXCurses.libs.TTY*), 25  
 Singleton (*class in GLXCurses.MainLoop*), 94  
 size\_column\_width (*GLXCurses.FileChooser.FileSelect attribute*), 83  
 size\_column\_width (*GLXCurses.FileSelect attribute*), 211  
 sizeof() (*in module GLXCurses.libs.Utils*), 31  
 skip\_pager\_hint (*GLXCurses.Window attribute*), 155  
 skip\_pager\_hint (*GLXCurses.Window.Window attribute*), 123  
 skip\_taskbar\_hint (*GLXCurses.Window attribute*), 155  
 skip\_taskbar\_hint (*GLXCurses.Window.Window attribute*), 123  
 sort\_by\_mtime (*GLXCurses.Window attribute*), 155

*Curses.lib.FileChooserFunctions.FileChooserUtils* style (*GLXCurses.Application* attribute), 166  
 attribute), 21 style (*GLXCurses.Application.Application* attribute),  
 sort\_by\_name (GLX- 43  
*Curses.lib.FileChooserFunctions.FileChooserUtils* style (*GLXCurses.Widget* attribute), 136  
 attribute), 20 style (*GLXCurses.Widget.Widget* attribute), 117  
 sort\_by\_size (GLX- subwin (*GLXCurses.Aera.Area* attribute), 42  
*Curses.lib.FileChooserFunctions.FileChooserUtils*  
 attribute), 20  
 sort\_mtime\_order (GLX- take\_focus (*GLXCurses.MenuShell* attribute), 170  
*Curses.lib.FileChooserFunctions.FileChooserUtils* take\_focus (*GLXCurses.MenuShell.MenuShell*  
 attribute), 21 attribute), 96  
 sort\_name\_order (GLX- tempo (*GLXCurses.Buzzer.Buzzer* attribute), 49  
*Curses.lib.FileChooserFunctions.FileChooserUtils* text (*GLXCurses.Button.Button* attribute), 48  
 attribute), 20 text (*GLXCurses.CheckButton* attribute), 157  
 sort\_size\_order (GLX- text (*GLXCurses.CheckButton.CheckButton* attribute),  
*Curses.lib.FileChooserFunctions.FileChooserUtils* 51  
 attribute), 20 text (*GLXCurses.Clipboard* attribute), 125  
 SortType (built-in variable), 218 text (*GLXCurses.Clipboards.Clipboard* attribute), 51  
 spacing (*GLXCurses.Box* attribute), 148 text (*GLXCurses.EntryBuffer* attribute), 183  
 spacing (*GLXCurses.Box.Box* attribute), 46 text (*GLXCurses.EntryBuffer.EntryBuffer* attribute), 78  
 spacing (*GLXCurses.MenuItem* attribute), 171 text (*GLXCurses.lib.TextUtils.TextUtils* attribute), 30  
 spacing (*GLXCurses.MenuItem.MenuItem* attribute), text (*GLXCurses.ProgressBar* attribute), 182  
 96 text (*GLXCurses.ProgressBar.ProgressBar* attribute),  
 Spot (class in *GLXCurses.lib.Spot*), 24 100  
 start() (*GLXCurses.MainLoop.MainLoop* method), text (*GLXCurses.RadioButton* attribute), 157  
 94 text (*GLXCurses.RadioButton.RadioButton* attribute),  
 startup\_id (*GLXCurses.Window* attribute), 155 100  
 startup\_id (*GLXCurses.Window.Window* attribute), text (*GLXCurses.TextBuffer* attribute), 146  
 123 text (*GLXCurses.TextBuffer.TextBuffer* attribute), 108  
 StateFlags (built-in variable), 217 text\_column (*GLXCurses.EntryCompletion* at-  
 StatusBar (class in *GLXCurses*), 171 tribute), 203  
 StatusBar (class in *GLXCurses.StatusBar*), 105 text\_column (GLX-  
 statusbar (*GLXCurses.Application* attribute), 166 *Curses.EntryCompletion.EntryCompletion*  
 statusbar (*GLXCurses.Application.Application* at- attribute), 81  
 tribute), 43 text\_index\_to\_layout\_index() (GLX-  
 stdscr (*GLXCurses.Aera.Area* attribute), 42 *Curses.Entry* method), 199  
 stdscr (*GLXCurses.lib.TTY.Screen* attribute), 25 text\_index\_to\_layout\_index() (GLX-  
 stdscr (*GLXCurses.Screen* attribute), 126 *Curses.Entry.Entry* method), 74  
 step\_increment (*GLXCurses.Adjustment* attribute), text\_short\_cut (*GLXCurses.MenuItem* attribute),  
 158 171  
 step\_increment (GLX- text\_short\_cut (*GLXCurses.MenuItem.MenuItem*  
*Curses.Adjustment.Adjustment* attribute), attribute), 96  
 37 text\_wrap() (*GLXCurses.lib.TextUtils.TextUtils*  
 stop() (*GLXCurses.MainLoop.MainLoop* method), 94 method), 30  
 store() (*GLXCurses.Clipboard* method), 126 TextAttributes (class in GLX-  
 store() (*GLXCurses.Clipboards.Clipboard* method), *Curses.lib.TextAttributes*), 28  
 52 TextBuffer (class in *GLXCurses*), 146  
 str (built-in variable), 213 TextBuffer (class in *GLXCurses.TextBuffer*), 108  
 strip\_hash() (*GLXCurses.Colors* static method), TextFonts (class in *GLXCurses.lib.TextFonts*), 29  
 130 TextTag (class in *GLXCurses*), 144  
 strip\_hash() (*GLXCurses.lib.Colors.Colors* static TextTag (class in *GLXCurses.TextTag*), 108  
 method), 18 TextTagTable (class in *GLXCurses*), 145  
 Style (class in *GLXCurses*), 131 TextTagTable (class in *GLXCurses.TextTagTable*),  
 Style (class in *GLXCurses.Style*), 107 110

- TextUtils (class in GLXCurses.libs.TextUtils), 29  
 TextView (class in GLXCurses), 146  
 TextView (class in GLXCurses.TextView), 110  
 timeout (GLXCurses.libs.TTY.Screen attribute), 27  
 timeout (GLXCurses.Screen attribute), 129  
 title (GLXCurses.Window attribute), 155  
 title (GLXCurses.Window.Window attribute), 123  
 to\_data() (GLXCurses.Image method), 212  
 to\_data() (GLXCurses.Image.Image method), 88  
 ToolBar (class in GLXCurses), 174  
 ToolBar (class in GLXCurses.ToolBar), 112  
 toolbar (GLXCurses.Application attribute), 166  
 toolbar (GLXCurses.Application.Application attribute), 43  
 ToolbarStyle (built-in variable), 218  
 tooltip\_text (GLXCurses.Widget attribute), 136  
 tooltip\_text (GLXCurses.Widget.Widget attribute), 117  
 top\_margin (GLXCurses.TextView attribute), 147  
 top\_margin (GLXCurses.TextView.TextView attribute), 112  
 touch\_screen() (GLXCurses.libs.TTY.Screen method), 28  
 touch\_screen() (GLXCurses.Screen method), 129  
 track\_visited\_links (GLXCurses.Label attribute), 177  
 track\_visited\_links (GLXCurses.Label.Label attribute), 89  
 transient\_for (GLXCurses.Window attribute), 155  
 transient\_for (GLXCurses.Window.Window attribute), 123  
 tuple (built-in variable), 213  
 type (GLXCurses.libs.ChildElement.ChildElement attribute), 16  
 type (GLXCurses.Window attribute), 155  
 type (GLXCurses.Window.Window attribute), 124  
 type\_hint (GLXCurses.Window attribute), 156  
 type\_hint (GLXCurses.Window.Window attribute), 124
- ## U
- unchild() (GLXCurses.Widget method), 139  
 unchild() (GLXCurses.Widget.Widget method), 120  
 unicode (built-in variable), 213  
 unparent() (GLXCurses.Widget method), 137  
 unparent() (GLXCurses.Widget.Widget method), 118  
 unset\_focus\_chain() (GLXCurses.Container method), 141  
 unset\_focus\_chain() (GLXCurses.Container.Container method), 55  
 unset\_invisible\_char() (GLXCurses.Entry method), 195  
 unset\_invisible\_char() (GLXCurses.Entry.Entry method), 71  
 up() (GLXCurses.libs.Group.Group method), 22  
 up() (GLXCurses.libs.Groups.Groups method), 23  
 update\_directory\_list() (GLXCurses.libs.FileChooserFunctions.FileChooserUtils method), 21  
 update\_preferred\_sizes() (GLXCurses.Button.Button method), 49  
 update\_preferred\_sizes() (GLXCurses.CheckButton method), 158  
 update\_preferred\_sizes() (GLXCurses.CheckButton.CheckButton method), 51  
 update\_preferred\_sizes() (GLXCurses.Dialog method), 165  
 update\_preferred\_sizes() (GLXCurses.Dialog.Dialog method), 60  
 update\_preferred\_sizes() (GLXCurses.Entry method), 202  
 update\_preferred\_sizes() (GLXCurses.Entry.Entry method), 77  
 update\_preferred\_sizes() (GLXCurses.FileChooser.FileSelect method), 83  
 update\_preferred\_sizes() (GLXCurses.FileSelect method), 211  
 update\_preferred\_sizes() (GLXCurses.Frame method), 169  
 update\_preferred\_sizes() (GLXCurses.Frame.Frame method), 86  
 update\_preferred\_sizes() (GLXCurses.HBox method), 152  
 update\_preferred\_sizes() (GLXCurses.HBox.HBox method), 86  
 update\_preferred\_sizes() (GLXCurses.HSeparator method), 182  
 update\_preferred\_sizes() (GLXCurses.HSeparator.HSeparator method), 87  
 update\_preferred\_sizes() (GLXCurses.Label method), 181  
 update\_preferred\_sizes() (GLXCurses.Label.Label method), 93  
 update\_preferred\_sizes() (GLXCurses.MenuBar method), 170  
 update\_preferred\_sizes() (GLXCurses.MenuBar.MenuBar method), 95  
 update\_preferred\_sizes() (GLXCurses.MessageBar method), 174  
 update\_preferred\_sizes() (GLXCurses.MessageBar.MessageBar method), 98  
 update\_preferred\_sizes() (GLXCurses.ProgressBar method), 182  
 update\_preferred\_sizes() (GLXCurses.ProgressBar.ProgressBar method), 98

- Curses.ProgressBar.ProgressBar* 100
- update\_preferred\_sizes()* (*GLXCurses.RadioButton* method), 157
- update\_preferred\_sizes()* (*GLXCurses.RadioButton.RadioButton* method), 100
- update\_preferred\_sizes()* (*GLXCurses.StatusBar* method), 172
- update\_preferred\_sizes()* (*GLXCurses.StatusBar.StatusBar* method), 107
- update\_preferred\_sizes()* (*GLXCurses.ToolBar* method), 174
- update\_preferred\_sizes()* (*GLXCurses.ToolBar.ToolBar* method), 113
- update\_preferred\_sizes()* (*GLXCurses.VBox* method), 151
- update\_preferred\_sizes()* (*GLXCurses.VBox.VBox* method), 113
- update\_preferred\_sizes()* (*GLXCurses.VSeparator* method), 182
- update\_preferred\_sizes()* (*GLXCurses.VSeparator.VSeparator* method), 113
- update\_preferred\_sizes()* (*GLXCurses.Window* method), 157
- update\_preferred\_sizes()* (*GLXCurses.Window.Window* method), 125
- update\_size()* (*GLXCurses.FileChooserMenu.FileChooserMenu* method), 84
- upper* (*GLXCurses.Adjustment* attribute), 158
- upper* (*GLXCurses.Adjustment.Adjustment* attribute), 37
- upper\_stepper\_sensitivity* (*GLXCurses.Range* attribute), 205
- upper\_stepper\_sensitivity* (*GLXCurses.Range.Range* attribute), 101
- urgency\_hint* (*GLXCurses.Window* attribute), 156
- urgency\_hint* (*GLXCurses.Window.Window* attribute), 124
- use\_markdown* (*GLXCurses.Label* attribute), 177
- use\_markdown* (*GLXCurses.Label.Label* attribute), 90
- use\_underline* (*GLXCurses.Label* attribute), 177
- use\_underline* (*GLXCurses.Label.Label* attribute), 90
- V**
- valign* (*GLXCurses.Widget* attribute), 136
- valign* (*GLXCurses.Widget.Widget* attribute), 117
- value* (*GLXCurses.Adjustment* attribute), 159
- value* (*GLXCurses.Adjustment.Adjustment* attribute), 37
- value* (*GLXCurses.ProgressBar* attribute), 182
- method*), *value* (*GLXCurses.ProgressBar.ProgressBar* attribute), 100
- VBox* (*class in GLXCurses*), 151
- VBox* (*class in GLXCurses.VBox*), 113
- vexpand* (*GLXCurses.Widget* attribute), 136
- vexpand* (*GLXCurses.Widget.Widget* attribute), 117
- vexpand\_set* (*GLXCurses.Widget* attribute), 136
- vexpand\_set* (*GLXCurses.Widget.Widget* attribute), 117
- visible* (*GLXCurses.Widget* attribute), 136
- visible* (*GLXCurses.Widget.Widget* attribute), 118
- VSeparator* (*class in GLXCurses*), 182
- VSeparator* (*class in GLXCurses.VSeparator*), 113
- W**
- wait\_for\_text()* (*GLXCurses.Clipboard* method), 126
- wait\_for\_text()* (*GLXCurses.Clipboards.Clipboard* method), 51
- Widget* (*class in GLXCurses*), 132
- Widget* (*class in GLXCurses.Widget*), 113
- widget* (*GLXCurses.libs.ChildElement.ChildElement* attribute), 15
- widget* (*GLXCurses.libs.Group.Group* attribute), 22
- widget* (*GLXCurses.libs.GroupElement.GroupElement* attribute), 22
- width* (*GLXCurses.Aera.Area* attribute), 41
- width* (*GLXCurses.libs.TextUtils.TextUtils* attribute), 30
- width\_chars* (*GLXCurses.Label* attribute), 177
- width\_chars* (*GLXCurses.Label.Label* attribute), 90
- width\_max* (*GLXCurses.Image* attribute), 211
- width\_max* (*GLXCurses.Image.Image* attribute), 87
- width\_max* (*GLXCurses.ImageConvert* attribute), 212
- width\_max* (*GLXCurses.libs.ImageConvert.ImageConvert* attribute), 23
- width\_original* (*GLXCurses.Image* attribute), 211
- width\_original* (*GLXCurses.Image.Image* attribute), 87
- width\_original* (*GLXCurses.ImageConvert* attribute), 212
- width\_original* (*GLXCurses.libs.ImageConvert.ImageConvert* attribute), 23
- width\_request* (*GLXCurses.Widget* attribute), 136
- width\_request* (*GLXCurses.Widget.Widget* attribute), 118
- Window* (*class in GLXCurses*), 152
- Window* (*class in GLXCurses.Window*), 120
- window* (*GLXCurses.Widget* attribute), 137
- window* (*GLXCurses.Widget.Widget* attribute), 118
- wrap* (*GLXCurses.Label* attribute), 178
- wrap* (*GLXCurses.Label.Label* attribute), 90
- wrap* (*GLXCurses.libs.TextUtils.TextUtils* attribute), 30
- wrap\_mode* (*GLXCurses.Label* attribute), 178



wrap\_mode (*GLXCurses.Label.Label* attribute), 90  
 wrap\_mode (*GLXCurses.libs.TextUtils.TextUtils* attribute), 30  
 wrap\_mode (*GLXCurses.TextView.TextView* attribute), 148  
 wrap\_mode (*GLXCurses.TextView.TextView* attribute), 112

## X

x (*GLXCurses.Aera.Area* attribute), 41  
 x\_offset (*GLXCurses.libs.Movable.Movable* attribute), 24  
 x\_pos\_history\_actual\_path (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 x\_pos\_history\_actual\_path (*GLXCurses.FileSelect* attribute), 210  
 x\_pos\_history\_actual\_path\_allowed\_size (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 x\_pos\_history\_actual\_path\_allowed\_size (*GLXCurses.FileSelect* attribute), 210  
 x\_pos\_history\_list\_label (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 x\_pos\_history\_list\_label (*GLXCurses.FileSelect* attribute), 210  
 x\_pos\_history\_next\_label (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 x\_pos\_history\_next\_label (*GLXCurses.FileSelect* attribute), 210  
 x\_pos\_history\_prev\_label (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 x\_pos\_history\_prev\_label (*GLXCurses.FileSelect* attribute), 210  
 x\_pos\_line\_start (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 x\_pos\_line\_start (*GLXCurses.FileSelect* attribute), 210  
 x\_pos\_line\_stop (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 x\_pos\_line\_stop (*GLXCurses.FileSelect* attribute), 210  
 x\_pos\_title\_mtime (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 x\_pos\_title\_mtime (*GLXCurses.FileSelect* attribute), 210  
 x\_pos\_title\_name (*GLXCurses.FileChooser.FileSelect* attribute), 83

x\_pos\_title\_name (*GLXCurses.FileSelect* attribute), 210  
 x\_pos\_title\_size (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 x\_pos\_title\_size (*GLXCurses.FileSelect* attribute), 210  
 xalign (*GLXCurses.Misc* attribute), 175  
 xalign (*GLXCurses.Misc.Misc* attribute), 98  
 xdg\_cache\_home (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory* attribute), 34  
 xdg\_config\_dirs (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory* attribute), 34  
 xdg\_config\_home (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory* attribute), 33  
 xdg\_data\_dirs (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory* attribute), 34  
 xdg\_data\_home (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory* attribute), 33  
 xdg\_runtime\_dir (*GLXCurses.libs.XDGBaseDirectory.XDGBaseDirectory* attribute), 34  
 XDGBaseDirectory (class in *GLXCurses.libs.XDGBaseDirectory*), 33  
 xpad (*GLXCurses.Misc* attribute), 175  
 xpad (*GLXCurses.Misc.Misc* attribute), 98

## Y

y (*GLXCurses.Aera.Area* attribute), 41  
 y\_offset (*GLXCurses.libs.Movable.Movable* attribute), 24  
 y\_pos\_history (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 y\_pos\_history (*GLXCurses.FileSelect* attribute), 210  
 y\_pos\_items (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 y\_pos\_items (*GLXCurses.FileSelect* attribute), 210  
 y\_pos\_titles (*GLXCurses.FileChooser.FileSelect* attribute), 83  
 y\_pos\_titles (*GLXCurses.FileSelect* attribute), 210  
 yalign (*GLXCurses.Misc* attribute), 175  
 yalign (*GLXCurses.Misc.Misc* attribute), 98  
 ypad (*GLXCurses.Misc* attribute), 175  
 ypad (*GLXCurses.Misc.Misc* attribute), 99