Qtile Documentation

Release 0.14.2

Aldo Cortesi

Feb 06, 2020

Contents

CHAPTER 1

Getting started

1.1 Installing Qtile

1.1.1 Distro Guides

Below are the preferred installation methods for specific distros. If you are running something else, please see *Installing From Source*.

Installing on Arch Linux

Stable versions of Qtile are currently packaged for Arch Linux. To install this package, run:

```
pacman -S qtile
```

Please see the ArchWiki for more information on Qtile.

Installing on Fedora

Stable versions of Qtile are currently packaged for current versions of Fedora. To install this package, run:

```
dnf -y install qtile
```

Installing on Funtoo

Latest versions of Qtile are available on Funtoo with Python 2.7, 3.4, and 3.5 implementations. To install it, run:

```
emerge -av x11-wm/qtile
```

You can also install the development version from GitHub:

```
echo "x11-wm/qtile-9999 **" >> /etc/portage/package.accept_keywords
emerge -av qtile
```

Customize

You can customize your installation with the following useflags:

- dbus
- widget-khal-calendar
- widget-imap
- widget-keyboardkbdd
- widget-launchbar
- widget-mpd
- widget-mpris
- widget-wlan

The dbus useflag is enabled by default. Disable it only if you know what it is and know you don't use/need it.

All widget-* useflags are disabled by default because these widgets require additional dependencies while not everyone will use them. Enable only widgets you need to avoid extra dependencies thanks to these useflags.

Visit Funtoo Qtile documentation for more details on Qtile installation on Funtoo.

Installing on Debian or Ubuntu

On recent Ubuntu (17.04 or greater) and Debian unstable versions, there are Qtile packages available via:

```
sudo apt-get install qtile
```

On older versions of Ubuntu (15.10 to 16.10) and Debian 9, the dependencies are available via:

```
sudo apt-get install python3-xcffib python3-cairocffi
```

Installing on Slackware

Qtile is available on the SlackBuilds.org as:

Package Name	Description
qtile	stable branch (release)

Using slpkg (third party package manager)

The easy way to install Qtile is with slpkg. For example:

```
slpkg -s sbo qtile
```

Manual installation

Download dependencies first and install them. The order in which you need to install is:

- pycparser
- cffi
- futures
- python-xcffib
- trollius
- cairocffi
- qtile

Please see the HOWTO for more information on SlackBuild Usage HOWTO.

Installing on FreeBSD

Qtile is available via FreeBSD Ports. It can be installed with

```
pkg install qtile
```

1.1.2 Installing From Source

First, you need to install all of Qtile's dependencies (although some are optional/not needed depending on your Python version, as noted below).

Note that Python 3 versions 3.5 and newer are currently supported and tested, including corresponding PyPy3 versions.

xcffib

Qtile uses xcffib as an XCB binding, which has its own instructions for building from source. However, if you'd like to skip building it, you can install its dependencies, you will need libxcb and libffi with the associated headers (`libxcb-render0-dev` and `libffi-dev` on Ubuntu), and install it via PyPI:

```
pip install xcffib
```

cairocffi

Qtile uses cairocffi with XCB support via xcffib. You'll need `libcairo2`, the underlying library used by the binding. You should **be sure before you install cairocffi that xcffib has been installed**, otherwise the needed cairo-xcb bindings will not be built. Once you've got the dependencies installed, you can use the latest version on PyPI:

```
pip install --no-cache-dir cairocffi
```

pangocairo

You'll also need `libpangocairo`, which on Ubuntu can be installed via `sudo apt-get install libpangocairo-1.0-0`. Qtile uses this to provide text rendering (and binds directly to it via cffi with a small in-tree binding).

dbus/gobject

Until someone comes along and writes an asyncio-based dbus library, qtile will depend on `python-dbus` to interact with dbus. This means that if you want to use things like notification daemon or mpris widgets, you'll need to install python-gobject and python-dbus. Qtile will run fine without these, although it will emit a warning that some things won't work.

Qtile

With the dependencies in place, you can now install qtile:

```
git clone git://github.com/qtile/qtile.git
cd qtile
pip install .
```

Stable versions of Qtile can be installed from PyPI:

```
pip install qtile
```

As long as the necessary libraries are in place, this can be done at any point, however, it is recommended that you first install xcffib to ensure the cairo-xcb bindings are built (see above).

The above steps are sufficient to run Qtile directly, but there are some extra works if you want to run it within a virtualenv. Here are the steps on a Fedora system for user `foo`, it should work on other Linux systems too.

1. Clone the repo as `~/local/qtile/`.

   ```
   mkdir -p ~/local/
   cd ~/local/
   git clone git://github.com/qtile/qtile.git
   ```

2. Create a virtualenv `~/local/qtile/venv/`, and install the dependencies there (see above).

3. Create a glue shell to take advantage of the virtualenv.

   ```
   cat > /home/foo/local/qtile/qtile-venv-entry <<EOF
   #!/bin/bash

   source ~/local/qtile/venv/bin/activate
   python ~/local/qtile/bin/qtile $*
   EOF
   ```

4. Create an xsession file. Note that it can only be used to log in as user `foo` due to file system permission restriction.

   ```
   cat > /usr/share/xsessions/qtile-venv.desktop <<EOF
   [Desktop Entry]
   Name=Qtile(venv)
   Comment=Qtile Session Within Venv
   Exec=/home/foo/local/qtile/qtile-venv-entry
   Type=Application
   Keywords=wm;tiling
   EOF
   ```

5. Log out or reboot your system, then select "Qtile(venv)" as your window manager by clicking the gear icon () when logging in again.

1.2 Configuration

Qtile is configured in Python. A script (`~/.config/qtile/config.py` by default) is evaluated, and a small set of configuration variables are pulled from its global namespace.

1.2.1 Configuration lookup order

Qtile looks in the following places for a configuration file, in order:

- The location specified by the `-c` argument.
- `$XDG_CONFIG_HOME/qtile/config.py`, if it is set
- `~/.config/qtile/config.py`
- It reads the module `libqtile.resources.default_config`, included by default with every Qtile installation.

Qtile will try to create the configuration file as a copy of the default config, if it doesn't exist yet.

1.2.2 Default Configuration

The default configuration is invoked when qtile cannot find a configuration file. In addition, if qtile is restarted via qshell, qtile will load the default configuration if the config file it finds has some kind of error in it. The documentation below describes the configuration lookup process, as well as what the key bindings are in the default config.

The default config is not intended to be suitable for all users; it's mostly just there so qtile does /something/ when fired up, and so that it doesn't crash and cause you to lose all your work if you reload a bad config.

Key Bindings

The mod key for the default config is `mod4`, which is typically bound to the "Super" keys, which are things like the windows key and the mac command key. The basic operation is:

- `mod + k` or `mod + j`: switch windows on the current stack
- `mod + <space>`: put focus on the other pane of the stack (when in stack layout)
- `mod + <tab>`: switch layouts
- `mod + w`: close window
- `mod + <ctrl> + r`: restart qtile with new config
- `mod + <group name>`: switch to that group
- `mod + <shift> + <group name>`: send a window to that group
- `mod + <enter>`: start xterm
- `mod + r`: start a little prompt in the bar so users can run arbitrary commands

The default config defines one screen and 8 groups, one for each letter in `asdfuiop`. It has a basic bottom bar that includes a group box, the current window name, a little text reminder that you're using the default config, a system tray, and a clock.

The default configuration has several more advanced key combinations, but the above should be enough for basic usage of qtile.

Mouse Bindings

By default, holding your `mod` key and clicking (and holding) a window will allow you to drag it around as a floating window.

1.2.3 Configuration variables

A Qtile configuration consists of a file with a bunch of variables in it, which qtile imports and then runs as a python file to derive its final configuration. The documentation below describes the most common configuration variables; more advanced configuration can be found in the qtile-examples repository, which includes a number of real-world configurations that demonstrate how you can tune Qtile to your liking. (Feel free to issue a pull request to add your own configuration to the mix!)

Lazy objects

The `lazy.lazy` object is a special helper object to specify a command for later execution. This object acts like the root of the object graph, which means that we can specify a key binding command with the same syntax used to call the command through a script or through *qshell*.

Example

```
from libqtile.config import Key
from libqtile.command import lazy

keys = [
    Key(
        ["mod1"], "k",
        lazy.layout.down()
    ),
    Key(
        ["mod1"], "j",
        lazy.layout.up()
    )
]
```

Lazy functions

This is overview of the commonly used functions for the key bindings. These functions can be called from commands on the *Qtile* object or on another object in the command tree.

Some examples are given below.

General functions

function	description
`lazy.spawn("application")`	Run the `application`
`lazy.spawncmd()`	Open command prompt on the bar. See prompt widget.
`lazy.restart()`	Restart Qtile and reload its config. It won't close your windows
`lazy.shutdown()`	Close the whole Qtile

Group functions

function	description
`lazy.next_layout()`	Use next layout on the actual group
`lazy.prev_layout()`	Use previous layout on the actual group
`lazy.screen.next_group()`	Move to the group on the right
`lazy.screen.prev_group()`	Move to the group on the left
`lazy.screen.toggle_group()`	Move to the last visited group
`lazy.group["group_name"].toscreen()`	Move to the group called `group_name`
`lazy.layout.increase_ratio()`	Increase the space for master window at the expense of slave windows
`lazy.layout.decrease_ratio()`	Decrease the space for master window in the advantage of slave windows

Window functions

function	description
`lazy.window.kill()`	Close the focused window
`lazy.layout.next()`	Switch window focus to other pane(s) of stack
`lazy.window.togroup("group_name")`	Move focused window to the group called `group_name`
`lazy.window.toggle_floating()`	Put the focused window to/from floating mode
`lazy.window.toggle_fullscreen()`	Put the focused window to/from fullscreen mode

ScratchPad DropDown functions

function	description
`lazy.group["group_name"].dropdown_toggle("name")`	Toggles the visibility of the specified DropDown window. On first use, the configured process is spawned.

Groups

A group is a container for a bunch of windows, analogous to workspaces in other window managers. Each client window managed by the window manager belongs to exactly one group. The `groups` config file variable should be initialized to a list of `DGroup` objects.

`DGroup` objects provide several options for group configuration. Groups can be configured to show and hide themselves when they're not empty, spawn applications for them when they start, automatically acquire certain groups, and various other options.

Example

```
from libqtile.config import Group, Match
groups = [
    Group("a"),
    Group("b"),
    Group("c", matches=[Match(wm_class=["Firefox"])]),
]

# allow mod3+1 through mod3+0 to bind to groups; if you bind your groups
# by hand in your config, you don't need to do this.
from libqtile.dgroups import simple_key_binder
dgroups_key_binder = simple_key_binder("mod3")
```

Reference

Group

class libqtile.config.**Group**(*name*, *matches=None*, *exclusive=False*, *spawn=None*, *layout=None*, *layouts=None*, *persist=True*, *init=True*, *layout_opts=None*, *screen_affinity=None*, *position=9223372036854775807*, *label=None*)

Represents a "dynamic" group

These groups can spawn apps, only allow certain Matched windows to be on them, hide when they're not in use, etc. Groups are identified by their name.

Parameters

name [string] the name of this group

matches [default `None`] list of `Match` objects whose windows will be assigned to this group

exclusive [boolean] when other apps are started in this group, should we allow them here or not?

spawn [string or list of strings] this will be `exec()` d when the group is created, you can pass either a program name or a list of programs to `exec()`

layout [string] the name of default layout for this group (e.g. 'max' or 'stack'). This is the name specified for a particular layout in config.py or if not defined it defaults in general the class name in all lower case.

layouts [list] the group layouts list overriding global layouts. Use this to define a separate list of layouts for this particular group.

persist [boolean] should this group stay alive with no member windows?

init [boolean] is this group alive when qtile starts?

position [int] group position

label [string] the display name of the group. Use this to define a display name other than name of the group. If set to None, the display name is set to the name.

`libqtile.dgroups.`**`simple_key_binder`**(*mod*, *keynames=None*)

Bind keys to mod+group position or to the keys specified as second argument

Group Matching

Match

`class` `libqtile.config.`**`Match`**(*title=None*, *wm_class=None*, *role=None*, *wm_type=None*, *wm_instance_class=None*, *net_wm_pid=None*)

Match for dynamic groups

It can match by title, class or role.

`Match` supports both regular expression objects (i.e. the result of `re.compile()`) or strings (match as a "include" match). If a window matches any of the things in any of the lists, it is considered a match.

Parameters

title: things to match against the title (WM_NAME)

wm_class: things to match against the second string in WM_CLASS atom

role: things to match against the WM_ROLE atom

wm_type: things to match against the WM_TYPE atom

wm_instance_class: things to match against the first string in WM_CLASS atom

net_wm_pid: things to match against the _NET_WM_PID atom (only int allowed in this rule)

Rule

`class` `libqtile.config.`**`Rule`**(*match*, *group=None*, *float=False*, *intrusive=False*, *break_on_match=True*)

How to act on a Match

A Rule contains a Match object, and a specification about what to do when that object is matched.

Parameters

match : `Match` object associated with this `Rule`

float : auto float this window?

intrusive : override the group's exclusive setting?

break_on_match : Should we stop applying rules if this rule is matched?

ScratchPad and DropDown

ScratchPad is a special - by default invisible - group which acts as a container for *DropDown* configurations. A *DropDown* can be configured to spawn a defined process and bind thats process' window to it. The associated window can then be shown and hidden by the lazy command `dropdown_toggle()` (see *Lazy objects*) from the ScratchPad group. Thus - for example - your favorite terminal emulator turns into a quake-like terminal by the control of qtile.

If the DropDown window turns visible it is placed as a floating window on top of the current group. If the DropDown is hidden, it is simply switched back to the ScratchPad group.

Example

```
from libqtile.config import Group, ScratchPad, DropDown, Key
from libqtile.command import lazy
groups = [
    ScratchPad("scratchpad", [
        # define a drop down terminal.
        # it is placed in the upper third of screen by default.
        DropDown("term", "urxvt", opacity=0.8),

        # define another terminal exclusively for qshell at different position
        DropDown("qshell", "urxvt -hold -e qshell",
                 x=0.05, y=0.4, width=0.9, height=0.6, opacity=0.9,
                 on_focus_lost_hide=True) ]),
    Group("a"),
]

keys = [
  # toggle visibiliy of above defined DropDown named "term"
  Key([], 'F11', lazy.group['scratchpad'].dropdown_toggle('term')),
  Key([], 'F12', lazy.group['scratchpad'].dropdown_toggle('qshell')),
]
```

There is only one DropDown visible in current group at a time. If a further DropDown is set visible the currently shown DropDown turns invisble immediately.

Note that if the window is set to not floating, it is detached from DropDown and ScratchPad, and a new pocess is spawned next time the DropDown is set visible.

Reference

ScratchPad

class libqtile.config.**ScratchPad**(*name*, *dropdowns=None*, *position=9223372036854775807*, *label=''*)

Represents a "ScratchPad" group

ScratchPad adds a (by default) invisible group to qtile. That group is used as a place for currently not visible windows spawned by a `DropDown` configuration.

Parameters

name [string] the name of this group

dropdowns [default `None`] list of DropDown objects

position [int] group position

label [string] The display name of the ScratchPad group. Defaults to the empty string such that the group is hidden in `GroupList` widget.

DropDown

class libqtile.config.**DropDown**(*name*, *cmd*, ***config*)

Configure a specified command and its associated window for the ScratchPad. That window can be shown and hidden using a configurable keystroke or any other scripted trigger.

key	default	description
`height`	`0.35`	Height of window as fraction of current screen.
`on_focus_lost_hide`	`True`	Shall the window be hidden if focus is lost? If so, the DropDown is hidden if window focus or the group is changed.
`opacity`	`0.9`	Opacity of window as fraction. Zero is opaque.
`warp_pointer`	`True`	Shall pointer warp to center of window on activation? This has only effect if any of the on_focus_lost_xxx configurations is True
`width`	`0.8`	Width of window as fraction of current screen width
`x`	`0.1`	X position of window as fraction of current screen width. 0 is the left most position.
`y`	`0.0`	Y position of window as fraction of current screen height. 0 is the top most position. To show the window at bottom, you have to configure a value < 1 and an appropriate height.

Keys

The `keys` variable defines Qtile's key bindings. Individual key bindings are defined with *`libqtile.config.Key`* as demonstrated in the following example. Note that you may specify more than one callback functions.

```
from libqtile.config import Key

keys = [
    # Pressing "Meta + Shift + a".
    Key(["mod4", "shift"], "a", callback, ...),

    # Pressing "Control + p".
    Key(["control"], "p", callback, ...),

    # Pressing "Meta + Tab".
    Key(["mod4", "mod1"], "Tab", callback, ...),
]
```

The above may also be written more concisely with the help of the `libqtile.config.EzKey` helper class. The following example is functionally equivalent to the above:

```
from libqtile.config import EzKey as Key

keys = [
```

(continues on next page)

(continued from previous page)

```
    Key("M-S-a", callback, ...),
    Key("C-p",   callback, ...),
    Key("M-A-<Tab>", callback, ...),
]
```

The `EzKey` modifier keys (i.e. `MASC`) can be overwritten through the `EzKey.modifier_keys` dictionary. The defaults are:

```
modifier_keys = {
    'M': 'mod4',
    'A': 'mod1',
    'S': 'shift',
    'C': 'control',
}
```

Modifiers

On most systems `mod1` is the Alt key - you can see which modifiers, which are enclosed in a list, map to which keys on your system by running the `xmodmap` command. This example binds `Alt-k` to the "down" command on the current layout. This command is standard on all the included layouts, and switches to the next window (where "next" is defined differently in different layouts). The matching "up" command switches to the previous window.

Modifiers include: "shift", "lock", "control", "mod1", "mod2", "mod3", "mod4", and "mod5". They can be used in combination by appending more than one modifier to the list:

```
Key(
    ["mod1", "control"], "k",
    lazy.layout.shuffle_down()
)
```

Special keys

These are most commonly used special keys. For complete list please see the code. You can create bindings on them just like for the regular keys. For example `Key(["mod1"], "F4", lazy.window.kill())`.

`Return`
`BackSpace`
`Tab`
`space`
`Home`, `End`
`Left`, `Up`, `Right`, `Down`
`F1`, `F2`, `F3`, ...
`XF86AudioRaiseVolume`
`XF86AudioLowerVolume`
`XF86AudioMute`
`XF86AudioNext`
`XF86AudioPrev`
`XF86MonBrightnessUp`
`XF86MonBrightnessDown`

Reference

Key

class libqtile.config.**Key**(*modifiers*, *key*, **commands*, ***kwargs*)
Defines a keybinding.

Parameters

modifiers: A list of modifier specifications. Modifier specifications are one of: "shift", "lock", "control", "mod1", "mod2", "mod3", "mod4", "mod5".

key: A key specification, e.g. "a", "Tab", "Return", "space".

commands: A list of lazy command objects generated with the lazy.lazy helper. If multiple Call objects are specified, they are run in sequence.

kwargs: A dictionary containing "desc", allowing a description to be added

EzConfig

class libqtile.config.**EzConfig**
Helper class for defining key and button bindings in an emacs-like format. Inspired by Xmonad's XMonad.Util.EZConfig.

Layouts

A layout is an algorithm for laying out windows in a group on your screen. Since Qtile is a tiling window manager, this usually means that we try to use space as efficiently as possible, and give the user ample commands that can be bound to keys to interact with layouts.

The `layouts` variable defines the list of layouts you will use with Qtile. The first layout in the list is the default. If you define more than one layout, you will probably also want to define key bindings to let you switch to the next and previous layouts.

See *Built-in Layouts* for a listing of available layouts.

Example

```
from libqtile import layout
layouts = [
    layout.Max(),
    layout.Stack(stacks=2)
]
```

Mouse

The `mouse` config file variable defines a set of global mouse actions, and is a list of *Click* and *Drag* objects, which define what to do when a window is clicked or dragged.

Example

```
from libqtile.config import Click, Drag
mouse = [
    Drag([mod], "Button1", lazy.window.set_position_floating(),
        start=lazy.window.get_position()),
    Drag([mod], "Button3", lazy.window.set_size_floating(),
        start=lazy.window.get_size()),
    Click([mod], "Button2", lazy.window.bring_to_front())
]
```

The above example can also be written more concisely with the help of the `EzClick` and `EzDrag` helpers:

```
from libqtile.config import EzClick as Click, EzDrag as Drag

mouse = [
    Drag("M-1", lazy.window.set_position_floating(),
        start=lazy.window.get_position()),
    Drag("M-3", lazy.window.set_size_floating(),
        start=lazy.window.get_size()),
    Click("M-2", lazy.window.bring_to_front())
]
```

Reference

Click

class libqtile.config.**Click**(*modifiers*, *button*, **commands*, ***kwargs*)

Defines binding of a mouse click

It focuses clicked window by default. If you want to prevent it, pass *focus=None* as an argument

Drag

class libqtile.config.**Drag**(**args*, *start=False*, ***kwargs*)

Defines binding of a mouse to some dragging action

On each motion event command is executed with two extra parameters added x and y offset from previous move

It focuses clicked window by default. If you want to prevent it pass, *focus=None* as an argument

Screens

The `screens` configuration variable is where the physical screens, their associated `bars`, and the `widgets` contained within the bars are defined.

See *Built-in Widgets* for a listing of available widgets.

Example

Tying together screens, bars and widgets, we get something like this:

```
from libqtile.config import Screen
from libqtile import bar, widget

screens = [
    Screen(
        bottom=bar.Bar([
            widget.GroupBox(),
            widget.WindowName()
            ], 30),
        ),
    Screen(
        bottom=bar.Bar([
            widget.GroupBox(),
            widget.WindowName()
            ], 30),
        )
    ]
```

Bars support both solid background colors and gradients by supplying a list of colors that make up a linear gradient. For example, `bar.Bar(..., background="#000000")` will give you a black back ground (the default), while `bar.Bar(..., background=["#000000", "#FFFFFF"])` will give you a background that fades from black to white.

Fake Screens

instead of using the variable *screens* the variable *fake_screens* can be used to set split a physical monitor into multiple screens. They can be used like this:

```
from libqtile.config import Screen
from libqtile import bar, widget

# screens look like this
#     600          300
#  |-------------|-----|
#  |           480|     |580
#  |   A          |  B  |
#  |----------|--|     |
#  |       400|--|-----|
#  |   C      |        |400
#  |----------|   D    |
#     500     |--------|
#                 400
#
# Notice there is a hole in the middle
# also D goes down below the others

fake_screens = [
  Screen(
      bottom=bar.Bar(
          [
              widget.Prompt(),
              widget.Sep(),
              widget.WindowName(),
              widget.Sep(),
              widget.Systray(),
              widget.Sep(),
```

(continues on next page)

(continued from previous page)

```
                widget.Clock(format='%H:%M:%S %d.%m.%Y')
            ],
            24,
            background="#555555"
        ),
        x=0,
        y=0,
        width=600,
        height=480
    ),
    Screen(
        top=bar.Bar(
            [
                widget.GroupBox(),
                widget.WindowName(),
                widget.Clock()
            ],
            30,
        ),
        x=600,
        y=0,
        width=300,
        height=580
    ),
    Screen(
        top=bar.Bar(
            [
                widget.GroupBox(),
                widget.WindowName(),
                widget.Clock()
            ],
            30,
        ),
        x=0,
        y=480,
        width=500,
        height=400
    ),
    Screen(
        top=bar.Bar(
            [
                widget.GroupBox(),
                widget.WindowName(),
                widget.Clock()
            ],
            30,
        ),
        x=500,
        y=580,
        width=400,
        height=400
    ),
]
```

Third-party bars

There might be some reasons to use third-party bars. For instance you can come from another window manager and you have already configured dzen2, xmobar, or something else. They definitely can be used with Qtile too. In fact, any additional configurations aren't needed. Just run the bar and qtile will adapt.

Reference

Screen

class libqtile.config.**Screen**(*top=None*, *bottom=None*, *left=None*, *right=None*, *x=None*, *y=None*, *width=None*, *height=None*)

A physical screen, and its associated paraphernalia.

Define a screen with a given set of Bars of a specific geometry. Note that bar.Bar objects can only be placed at the top or the bottom of the screen (bar.Gap objects can be placed anywhere). Also, `x`, `y`, `width`, and `height` aren't specified usually unless you are using 'fake screens'.

Parameters

top: Gap/Bar object, or None.

bottom: Gap/Bar object, or None.

left: Gap/Bar object, or None.

right: Gap/Bar object, or None.

x [int or None]

y [int or None]

width [int or None]

height [int or None]

Bar

class libqtile.bar.**Bar**(*widgets*, *size*, ***config*)

A bar, which can contain widgets

Parameters

widgets : A list of widget objects.

size : The "thickness" of the bar, i.e. the height of a horizontal bar, or the width of a vertical bar.

key	default	description
`background`	`'#000000'`	Background colour.
`opacity`	`1`	Bar window opacity.

Gap

class libqtile.bar.**Gap**(*size*)

A gap placed along one of the edges of the screen

If a gap has been defined, Qtile will avoid covering it with windows. The most probable reason for configuring a gap is to make space for a third-party bar or other static window.

Parameters

size : The "thickness" of the gap, i.e. the height of a horizontal gap, or the width of a vertical gap.

Hooks

Qtile provides a mechanism for subscribing to certain events in `libqtile.hook`. To subscribe to a hook in your configuration, simply decorate a function with the hook you wish to subscribe to.

See *Built-in Hooks* for a listing of available hooks.

Examples

Automatic floating dialogs

Let's say we wanted to automatically float all dialog windows (this code is not actually necessary; Qtile floats all dialogs by default). We would subscribe to the `client_new` hook to tell us when a new window has opened and, if the type is "dialog", as can set the window to float. In our configuration file it would look something like this:

```
from libqtile import hook

@hook.subscribe.client_new
def floating_dialogs(window):
    dialog = window.window.get_wm_type() == 'dialog'
    transient = window.window.get_wm_transient_for()
    if dialog or transient:
        window.floating = True
```

A list of available hooks can be found in the *Built-in Hooks* reference.

Autostart

If you want to run commands or spawn some applications when Qtile starts, you'll want to look at the `startup` and `startup_once` hooks. `startup` is emitted every time Qtile starts (including restarts), whereas `startup_once` is only emitted on the very first startup.

Let's create a file `~/.config/qtile/autostart.sh` that will set our desktop wallpaper and start a few programs when Qtile first runs.

```
#!/bin/sh
feh --bg-scale ~/images/wallpaper.jpg &
pidgin &
dropbox start &
```

We can then subscribe to `startup_once` to run this script:

```
import os
import subprocess

@hook.subscribe.startup_once
```

(continues on next page)

(continued from previous page)

```
def autostart():
    home = os.path.expanduser('~/.config/qtile/autostart.sh')
    subprocess.call([home])
```

In addition to the above variables, there are several other boolean configuration variables that control specific aspects of Qtile's behavior:

variable	default	description
auto_fullscreen	True	If a window requests to be fullscreen, it is automatically fullscreened. Set this to false if you only want windows to be fullscreen if you ask them to be.
bring_front_click	False	When clicked, should the window be brought to the front or not. (This sets the X Stack Mode to Above.)
cursor_warp	False	If true, the cursor follows the focus as directed by the keyboard, warping to the center of the focused window.
dgroups_key_binder	None	A function which generates group binding hotkeys. It takes a single argument, the DGroups object, and can use that to set up dynamic key bindings. A sample implementation is available in libqtile/dgroups.py called simple_key_binder(), which will bind groups to mod+shift+0-10 by default.
dgroups_app_rules	[]	A list of Rule objects which can send windows to various groups based on matching criteria.
extension_defaults	copy of *widget_defaults*	Default settings for extensions.
floating_layout	layout.Floating(float_rules=[...])	The default floating layout to use. This allows you to set custom floating rules among other things if you wish. See the configuration file for the default *float_rules*.
focus_on_window_activation	smart	Behavior of the _NET_ACTIVATE_WINDOW message sent by applications • urgent: urgent flag is set for the window • focus: automatically focus the window • smart: automatically focus if the window is in the current group
follow_mouse_focus	True	Controls whether or not focus follows the mouse around as it moves across windows in a layout.
main	None	This is a function which takes one argument, the qtile object, and is run after the qtile object has been initialized. This allows people to monkey patch in any code they want to before qtile actually starts. Note that we only consider the config objects and hooks to be public APIs, and reserve the right to refactor the internals at any time. Use at your own risk!
widget_defaults	**dict(font='sans',** fontsize=12, padding=3)	Default settings for bar widgets.
wmname	"LG3D"	Gasp! We're lying here. In fact, nobody really uses or cares about this string besides java UI toolkits; you can see several discussions on the mailing lists, GitHub issues, and other WM documentation that suggest setting this string if your java app doesn't work correctly. We may as well just lie and say that we're a working one by default. We choose LG3D to maximize irony: it is a 3D non-reparenting WM written in java that happens to be on java's whitelist.

1.2.4 Testing your configuration

The best way to test changes to your configuration is with the provided Xephyr script. This will run Qtile with your `config.py` inside a nested X server and prevent your running instance of Qtile from crashing if something goes

wrong.

See *Hacking Qtile* for more information on using Xephyr.

1.2.5 Starting Qtile

There are several ways to start Qtile. The most common way is via an entry in your X session manager's menu. The default Qtile behavior can be invoked by creating a qtile.desktop file in `/usr/share/xsessions`.

A second way to start Qtile is a custom X session. This way allows you to invoke Qtile with custom arguments, and also allows you to do any setup you want (e.g. special keyboard bindings like mapping caps lock to control, setting your desktop background, etc.) before Qtile starts. If you're using an X session manager, you still may need to create a `custom.desktop` file similar to the `qtile.desktop` file above, but with `Exec=/etc/X11/xsession`. Then, create your own `~/.xsession`. There are several examples of user defined `xsession` s in the qtile-examples repository.

Finally, if you're a gnome user, you can start integrate Qtile into Gnome's session manager and use gnome as usual:

Running Inside Gnome

Add the following snippet to your Qtile configuration. As per this page, it registers Qtile with gnome-session. Without it, a "Something has gone wrong!" message shows up a short while after logging in. dbus-send must be on your $PATH.

```
import subprocess
import os
from libqtile import hook

@hook.subscribe.startup
def dbus_register():
    id = os.environ.get('DESKTOP_AUTOSTART_ID')
    if not id:
        return
    subprocess.Popen(['dbus-send',
                      '--session',
                      '--print-reply',
                      '--dest=org.gnome.SessionManager',
                      '/org/gnome/SessionManager',
                      'org.gnome.SessionManager.RegisterClient',
                      'string:qtile',
                      'string:' + id])
```

This adds a new entry "Qtile GNOME" to GDM's login screen.

```
$ cat /usr/share/xsessions/qtile_gnome.desktop
[Desktop Entry]
Name=Qtile GNOME
Comment=Tiling window manager
TryExec=/usr/bin/gnome-session
Exec=gnome-session --session=qtile
Type=XSession
```

The custom session for gnome-session.

For Gnome >= 3.23.2 (Ubuntu >= 17.04, Fedora >= 26, etc.)

```
$ cat /usr/share/gnome-session/sessions/qtile.session
[GNOME Session]
Name=Qtile session
RequiredComponents=qtile;org.gnome.SettingsDaemon.A11ySettings;org.gnome.
↪SettingsDaemon.Clipboard;org.gnome.SettingsDaemon.Color;org.gnome.SettingsDaemon.
↪Datetime;org.gnome.SettingsDaemon.Housekeeping;org.gnome.SettingsDaemon.Keyboard;
↪org.gnome.SettingsDaemon.MediaKeys;org.gnome.SettingsDaemon.Mouse;org.gnome.
↪SettingsDaemon.Power;org.gnome.SettingsDaemon.PrintNotifications;org.gnome.
↪SettingsDaemon.Rfkill;org.gnome.SettingsDaemon.ScreensaverProxy;org.gnome.
↪SettingsDaemon.Sharing;org.gnome.SettingsDaemon.Smartcard;org.gnome.SettingsDaemon.
↪Sound;org.gnome.SettingsDaemon.Wacom;org.gnome.SettingsDaemon.XSettings;
```

Or for older Gnome versions

```
$ cat /usr/share/gnome-session/sessions/qtile.session
[GNOME Session]
Name=Qtile session
RequiredComponents=qtile;gnome-settings-daemon;
```

So that Qtile starts automatically on login.

```
$ cat /usr/share/applications/qtile.desktop
[Desktop Entry]
Type=Application
Encoding=UTF-8
Name=Qtile
Exec=qtile
NoDisplay=true
X-GNOME-WMName=Qtile
X-GNOME-Autostart-Phase=WindowManager
X-GNOME-Provides=windowmanager
X-GNOME-Autostart-Notify=false
```

The above does not start gnome-panel. Getting gnome-panel to work requires some extra Qtile configuration, mainly making the top and bottom panels static on panel startup and leaving a gap at the top (and bottom) for the panel window.

You might want to add keybindings to log out of the GNOME session.

```
Key([mod, 'control'], 'l', lazy.spawn('gnome-screensaver-command -l')),
Key([mod, 'control'], 'q', lazy.spawn('gnome-session-quit --logout --no-prompt')),
Key([mod, 'shift', 'control'], 'q', lazy.spawn('gnome-session-quit --power-off')),
```

The above apps need to be in your path (though they are typically installed in `/usr/bin`, so they probably are if they're installed at all).

CHAPTER 2

Commands and scripting

2.1 Commands API

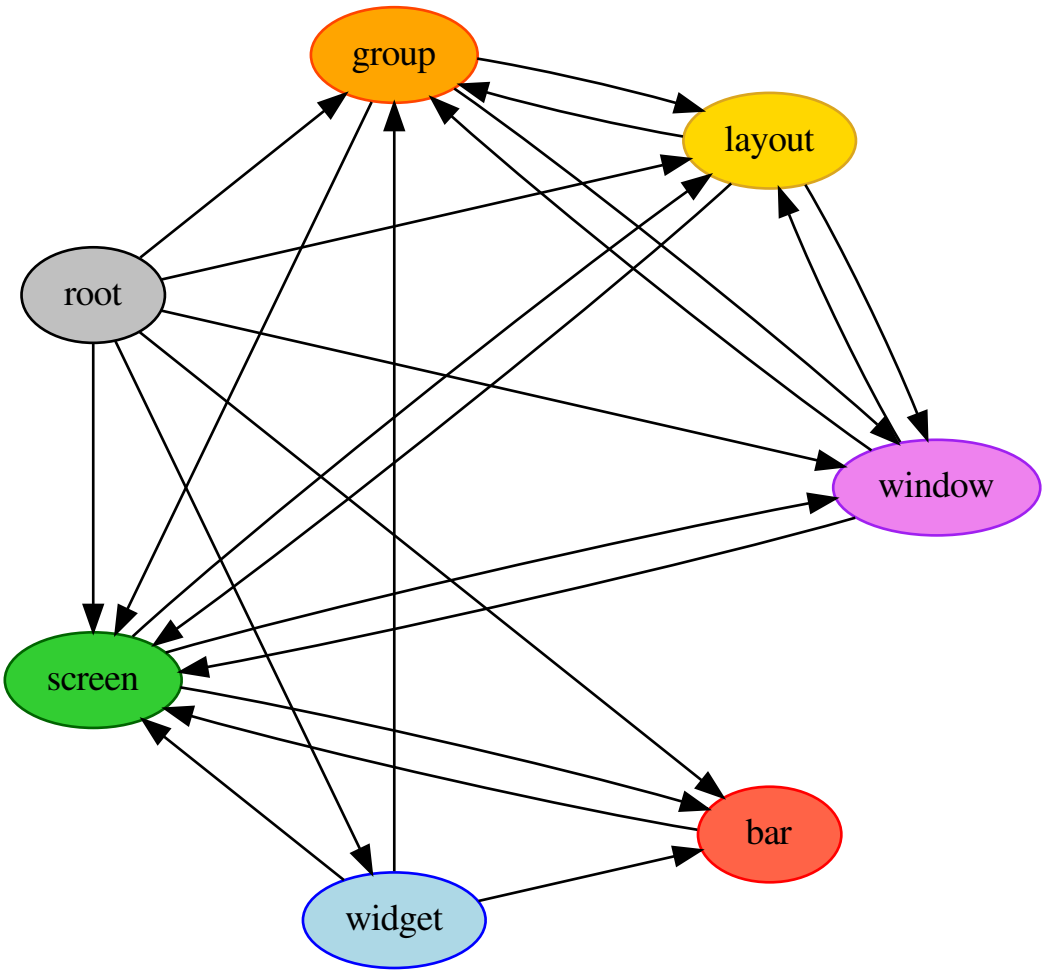
Qtile's command API is based on a graph of objects, where each object has a set of associated commands. The graph and object commands are used in a number of different places:

- Commands can be *bound to keys* in the Qtile configuration file.
- Commands can be *called through qshell*, the Qtile shell.
- The qsh can also be hooked into a Jupyter kernel *called iqshell*.
- Commands can be *called from a script* to interact with Qtile from Python.

If the explanation below seems a bit complex, please take a moment to explore the API using the `qshell` command shell. Command lists and detailed documentation can be accessed from its built-in help command.

2.1.1 Object Graph

The objects in Qtile's object graph come in seven flavours, matching the seven basic components of the window manager: `layouts`, `windows`, `groups`, `bars`, `widgets`, `screens`, and a special `root` node. Objects are addressed by a path specification that starts at the root, and follows the edges of the graph. This is what the graph looks like:



Each arrow can be read as "holds a reference to". So, we can see that a `widget` object *holds a reference to* objects of type `bar`, `screen` and `group`. Lets start with some simple examples of how the addressing works. Which particular objects we hold reference to depends on the context - for instance, widgets hold a reference to the screen that they appear on, and the bar they are attached to.

Lets look at an example, starting at the root node. The following script runs the `status` command on the root node, which, in this case, is represented by the Client object:

```
from libqtile.command import Client
c = Client()
print c.status()
```

From the graph, we can see that the root node holds a reference to `group` nodes. We can access the "info" command on the current group like so:

```
c.group.info()
```

To access a specific group, regardless of whether or not it is current, we use the Python containment syntax. This

command sends group "b" to screen 1 (by the `libqtile.config.Group.to_screen()` method):

```
c.group["b"].to_screen(1)
```

The current `group`, `layout`, `screen` and `window` can be accessed by simply leaving the key specifier out. The key specifier is mandatory for `widget` and `bar` nodes.

We can now drill down deeper in the graph. To access the screen currently displaying group "b", we can do this:

```
c.group["b"].screen.info()
```

Be aware, however, that group "b" might not currently be displayed. In that case, it has no associated screen, the path resolves to a non-existent node, and we get an exception:

```
libqtile.command.CommandError: No object screen in path 'group['b'].screen'
```

The graph is not a tree, since it can contain cycles. This path (redundantly) specifies the group belonging to the screen that belongs to group "b":

```
c.group["b"].screen.group
```

2.1.2 Keys

The key specifier for the various object types are as follows:

Object	Key	Optional?	Example
bar	"top", "bottom"	No	c.screen.bar["bottom"]
group	Name string	Yes	c.group["one"] c.group
layout	Integer index	Yes	c.layout[2] c.layout
screen	Integer index	Yes	c.screen[1] c.screen
widget	Widget name	No	c.widget["textbox"]
window	Integer window ID	Yes	c.window[123456] c.window

2.2 Scripting

2.2.1 Client-Server Scripting Model

Qtile has a client-server control model - the main Qtile instance listens on a named pipe, over which marshalled command calls and response data is passed. This allows Qtile to be controlled fully from external scripts. Remote interaction occurs through an instance of the `libqtile.command.Client` class. This class establishes a connection to the currently running instance of Qtile, and sources the user's configuration file to figure out which commands should be exposed. Commands then appear as methods with the appropriate signature on the `Client` object. The object hierarchy is described in the *Commands API* section of this manual. Full command documentation is available through the *Qtile Shell*.

2.2.2 Example

Below is a very minimal example script that inspects the current qtile instance, and returns the integer offset of the current screen.

```
from libqtile.command import Client
c = Client()
print c.screen.info()["index"]
```

2.3 qshell

The Qtile command shell is a command-line shell interface that provides access to the full complement of Qtile command functions. The shell features command name completion, and full command documentation can be accessed from the shell itself. The shell uses GNU Readline when it's available, so the interface can be configured to, for example, obey VI keybindings with an appropriate `.inputrc` file. See the GNU Readline documentation for more information.

2.3.1 Navigating the Object Graph

The shell presents a filesystem-like interface to the object graph - the builtin "cd" and "ls" commands act like their familiar shell counterparts:

```
> ls
layout/  widget/  screen/  bar/     window/  group/

> cd bar

bar> ls
bottom/

bar> cd bottom

bar['bottom']> ls
screen/

bar['bottom']> cd ../..

> ls
layout/  widget/  screen/  bar/     window/  group/
```

Note that the shell provides a "short-hand" for specifying node keys (as opposed to children). The following is a valid shell path:

```
> cd group/4/window/31457314
```

The command prompt will, however, always display the Python node path that should be used in scripts and key bindings:

```
group['4'].window[31457314]>
```

2.3.2 Live Documentation

The shell `help` command provides the canonical documentation for the Qtile API:

```
> cd layout/1

layout[1]> help
help command   -- Help for a specific command.

Builtins
========
cd    exit  help  ls    q     quit

Commands for this object
========================
add             commands      current       delete        doc
down            get info      items         next          previous
rotate          shuffle_down  shuffle_up    toggle_split  up

layout[1]> help previous
previous()
Focus previous stack.
```

2.3.3 Reference

Qsh

class libqtile.sh.**QSh**(*client: libqtile.command_interface.CommandInterface*, *completekey='tab'*)

Qtile shell instance

do_cd(*self*, *arg*) → str

Change to another path.

Examples

cd layout/0

cd ../layout

do_exit(*self*, *args*) → None

Exit qshell

do_ls(*self*, *arg: str*) → str

List contained items on a node.

Examples

> ls > ls ../layout

do_pwd(*self*, *arg*) → str
Returns the current working location

This is the same information as presented in the qshell prompt, but is very useful when running iqshell.

Examples

> pwd / > cd bar/top bar['top']> pwd bar['top']

do_help(*self*, *arg*) → str
Give help on commands and builtins

When invoked without arguments, provides an overview of all commands. When passed as an argument, also provides a detailed help on a specific command or builtin.

Examples

> help

> help command

2.4 iqshell

In addition to the standard `qshell` shell interface, we provide a kernel capable of running through Jupyter that hooks into the qshell client. The command structure and syntax is the same as qshell, so it is recommended you read that for more information about that.

2.4.1 Dependencies

In order to run iqshell, you must have ipykernel and jupyter_console. You can install the dependencies when you are installing qtile by running:

```
$ pip install qtile[ipython]
```

Otherwise, you can just install these two packages separately, either through PyPI or through your distribution package manager.

2.4.2 Installing and Running the Kernel

Once you have the required dependencies, you can run the kernel right away by running:

```
$ python -m libqtile.interactive.iqshell_kernel
```

However, this will merely spawn a kernel instance, you will have to run a separate frontend that connects to this kernel.

A more convenient way to run the kernel is by registering the kernel with Jupyter. To register the kernel itself, run:

```
$ python -m libqtile.interactive.iqshell_install
```

If you run this as a non-root user, or pass the `--user` flag, this will install to the user Jupyter kernel directory. You can now invoke the kernel directly when starting a Jupyter frontend, for example:

```
$ jupyter console --kernel qshell
```

The `iqshell` script will launch a Jupyter terminal console with the qshell kernel.

2.4.3 iqshell vs qshell

One of the main drawbacks of running through a Jupyter kernel is the frontend has no way to query the current node of the kernel, and as such, there is no way to set a custom prompt. In order to query your current node, you can call `pwd`.

This, however, enables many of the benefits of running in a Jupyter frontend, including being able to save, run, and re-run code cells in frontends such as the Jupyter notebook.

The Jupyter kernel also enables more advanced help, text completion, and introspection capabilities (however, these are currently not implemented at a level much beyond what is available in the standard qshell).

2.5 qtile-top

Is a top like to measure memory usage of qtile's internals.

2.6 qtile-run

Run a command applying rules to the new windows, ie, you can start a window in a specific group, make it floating, intrusive, etc.

The Windows must have NET_WM_PID.

```
# run xterm floating on group "test-group"
qtile-run -g test-group -f xterm
```

2.7 qtile-cmd

This is a simple tool to expose qtile.command functionality to shell. This can be used standalone or in other shell scripts.

2.7.1 Examples:

Output of `qtile-cmd -h`

```
usage: qtile-cmd [-h] [--object OBJ_SPEC [OBJ_SPEC ...]]
                 [--function FUNCTION] [--args ARGS [ARGS ...]] [--info]

Simple tool to expose qtile.command functionality to shell.

optional arguments:
```

(continues on next page)

(continued from previous page)

```
  -h, --help            show this help message and exit
  --object OBJ_SPEC [OBJ_SPEC ...], -o OBJ_SPEC [OBJ_SPEC ...]
                        Specify path to object (space separated). If no
                        --function flag display available commands.
  --function FUNCTION, -f FUNCTION
                        Select function to execute.
  --args ARGS [ARGS ...], -a ARGS [ARGS ...]
                        Set arguments supplied to function.
  --info, -i            With both --object and --function args prints
                        documentation for function.

Examples:
 qtile-cmd
 qtile-cmd -o cmd
 qtile-cmd -o cmd -f prev_layout -i
 qtile-cmd -o cmd -f prev_layout -a 3 # prev_layout on group 3
 qtile-cmd -o group 3 -f focus_back
```

Output of `qtile-cmd -o group 3`

```
-o group 3 -f commands            Returns a list of possible commands for this object
-o group 3 -f doc               * Returns the documentation for a specified command␣
↪name
-o group 3 -f eval              * Evaluates code in the same context as this function
-o group 3 -f focus_back          Focus the window that had focus before the current␣
↪one got it.
-o group 3 -f focus_by_name     * Focus the first window with the given name. Do␣
↪nothing if the name is
-o group 3 -f function          * Call a function with current object as argument
-o group 3 -f info                Returns a dictionary of info for this group
-o group 3 -f info_by_name      * Get the info for the first window with the given␣
↪name without giving it
-o group 3 -f items             * Returns a list of contained items for the specified␣
↪name
-o group 3 -f next_window         Focus the next window in group.
-o group 3 -f prev_window         Focus the previous window in group.
-o group 3 -f set_label         * Set the display name of current group to be used in␣
↪GroupBox widget.
-o group 3 -f setlayout
-o group 3 -f switch_groups     * Switch position of current group with name
-o group 3 -f toscreen          * Pull a group to a specified screen.
-o group 3 -f unminimize_all      Unminimise all windows in this group
```

Output of `qtile-cmd -o cmd`

```
-o cmd -f add_rule              * Add a dgroup rule, returns rule_id needed to remove␣
↪it
-o cmd -f addgroup              * Add a group with the given name
-o cmd -f commands                Returns a list of possible commands for this object
-o cmd -f critical                Set log level to CRITICAL
-o cmd -f debug                   Set log level to DEBUG
-o cmd -f delgroup              * Delete a group with the given name
-o cmd -f display_kb            * Display table of key bindings
```

(continues on next page)

(continued from previous page)

```
-o cmd -f doc                    * Returns the documentation for a specified command␣
↪name
-o cmd -f error                    Set log level to ERROR
-o cmd -f eval                   * Evaluates code in the same context as this function
-o cmd -f findwindow             * Launch prompt widget to find a window of the given␣
↪name
-o cmd -f focus_by_click         * Bring a window to the front
-o cmd -f function               * Call a function with current object as argument
-o cmd -f get_info                 Prints info for all groups
-o cmd -f get_state                Get pickled state for restarting qtile
-o cmd -f get_test_data            Returns any content arbitrarily set in the self.
↪test_data attribute.
-o cmd -f groups                   Return a dictionary containing information for all␣
↪groups
-o cmd -f hide_show_bar          * Toggle visibility of a given bar
-o cmd -f info                     Set log level to INFO
-o cmd -f internal_windows         Return info for each internal window (bars, for␣
↪example)
-o cmd -f items                  * Returns a list of contained items for the specified␣
↪name
-o cmd -f list_widgets             List of all addressible widget names
-o cmd -f next_layout            * Switch to the next layout.
-o cmd -f next_screen              Move to next screen
-o cmd -f next_urgent              Focus next window with urgent hint
-o cmd -f pause                    Drops into pdb
-o cmd -f prev_layout            * Switch to the previous layout.
-o cmd -f prev_screen              Move to the previous screen
-o cmd -f qtile_info               Returns a dictionary of info on the Qtile instance
-o cmd -f qtilecmd               * Execute a Qtile command using the client syntax
-o cmd -f remove_rule            * Remove a dgroup rule by rule_id
-o cmd -f restart                  Restart qtile
-o cmd -f run_extension          * Run extensions
-o cmd -f run_extention          * Deprecated alias for cmd_run_extension()
-o cmd -f run_external           * Run external Python script
-o cmd -f screens                  Return a list of dictionaries providing information␣
↪on all screens
-o cmd -f shutdown                 Quit Qtile
-o cmd -f simulate_keypress      * Simulates a keypress on the focused window.
-o cmd -f spawn                  * Run cmd in a shell.
-o cmd -f spawncmd               * Spawn a command using a prompt widget, with tab-
↪completion.
-o cmd -f status                   Return "OK" if Qtile is running
-o cmd -f switch_groups          * Switch position of groupa to groupb
-o cmd -f switchgroup            * Launch prompt widget to switch to a given group to␣
↪the current screen
-o cmd -f sync                     Sync the X display. Should only be used for␣
↪development
-o cmd -f to_layout_index        * Switch to the layout with the given index in self.
↪layouts.
-o cmd -f to_screen              * Warp focus to screen n, where n is a 0-based screen␣
↪number
-o cmd -f togroup                * Launch prompt widget to move current window to a␣
↪given group
-o cmd -f tracemalloc_dump         Dump tracemalloc snapshot
-o cmd -f tracemalloc_toggle       Toggle tracemalloc status
-o cmd -f warning                  Set log level to WARNING
-o cmd -f windows                  Return info for each client window
```

2.8 dqtile-cmd

A Rofi/dmenu interface to qtile-cmd. Accepts all arguments of qtile-cmd.

2.8.1 Examples:

Output of `dqtile-cmd -o cmd`

```
dmenu:                                                                    -
Alt-1   Prompt for args and show function help (if -f is present)
..      Go back to menu.
C-u     Clear input
Esc     Exit
-o cmd -f add_rule              * Add a dgroup rule, returns rule_id needed to remove it
-o cmd -f addgroup              * Add a group with the given name
-o cmd -f commands                Returns a list of possible commands for this object
-o cmd -f critical                Set log level to CRITICAL
-o cmd -f debug                   Set log level to DEBUG
-o cmd -f delgroup              * Delete a group with the given name
-o cmd -f display_kb            * Display table of key bindings
-o cmd -f doc                   * Returns the documentation for a specified command name
-o cmd -f error                   Set log level to ERROR
-o cmd -f eval                  * Evaluates code in the same context as this function
-o cmd -f findwindow            * Launch prompt widget to find a window of the given name
-o cmd -f focus_by_click        * Bring a window to the front
-o cmd -f function              * Call a function with current object as argument
-o cmd -f get_info                Prints info for all groups
-o cmd -f get_state               Get pickled state for restarting qtile
```

Output of `dqtile-cmd -h`

```
dqtile-cmd

    A Rofi/dmenu interface to qtile-cmd. Excepts all arguments of qtile-cmd
    (see below).

usage: dqtile-cmd [-h] [--object OBJ_SPEC [OBJ_SPEC ...]]
                  [--function FUNCTION] [--args ARGS [ARGS ...]] [--info]

Simple tool to expose qtile.command functionality to shell.

optional arguments:
  -h, --help            show this help message and exit
  --object OBJ_SPEC [OBJ_SPEC ...], -o OBJ_SPEC [OBJ_SPEC ...]
                        Specify path to object (space separated). If no
                        --function flag display available commands.
  --function FUNCTION, -f FUNCTION
                        Select function to execute.
  --args ARGS [ARGS ...], -a ARGS [ARGS ...]
                        Set arguments supplied to function.
  --info, -i            With both --object and --function args prints
                        documentation for function.
```

(continues on next page)

(continued from previous page)

```
Examples:
 dqtile-cmd
 dqtile-cmd -o cmd
 dqtile-cmd -o cmd -f prev_layout -i
 dqtile-cmd -o cmd -f prev_layout -a 3 # prev_layout on group 3
 dqtile-cmd -o group 3 -f focus_back

If both rofi and dmenu are present rofi will be selected as default, to change this␣
↪us --force-dmenu as the first argument.
```

CHAPTER 3

Getting involved

3.1 Contributing

3.1.1 Reporting bugs

Perhaps the easiest way to contribute to Qtile is to report any bugs you run into on the GitHub issue tracker.

Useful bug reports are ones that get bugs fixed. A useful bug report normally has two qualities:

1. **Reproducible.** If your bug is not reproducible it will never get fixed. You should clearly mention the steps to reproduce the bug. Do not assume or skip any reproducing step. Described the issue, step-by-step, so that it is easy to reproduce and fix.
2. **Specific.** Do not write a essay about the problem. Be Specific and to the point. Try to summarize the problem in minimum words yet in effective way. Do not combine multiple problems even they seem to be similar. Write different reports for each problem.

3.1.2 Writing code

To get started writing code for Qtile, check out our guide to *Hacking on Qtile*.

Submit a pull request

You've done your hacking and are ready to submit your patch to Qtile. Great! Now it's time to submit a pull request to our issue tracker on GitHub.

Important: Pull requests are not considered complete until they include all of the following:

- **Code** that conforms to PEP8.
- **Unit tests** that pass locally and in our CI environment.
- **Documentation** updates on an as needed basis.

Feel free to add your contribution (no matter how small) to the appropriate place in the CHANGELOG as well!

3.2 Hacking on Qtile

3.2.1 Requirements

Any reasonably recent version of these should work, so you can probably just install them from your package manager.

- pytest
- Xephyr
- `xrandr`, `xcalc`, `xeyes` and `xclock` (`x11-apps` on Ubuntu)

On Ubuntu, if testing on Python 3, this can be done with:

```
sudo apt-get install python3-pytest xserver-xephyr x11-apps
```

On ArchLinux, the X11 requirements are installed with:

```
sudo pacman -S xorg-xrandr xorg-xcalc xorg-xeyes xorg-xclock
```

3.2.2 Building cffi module

Qtile ships with a small in-tree pangocairo binding built using cffi, `pangocffi.py`, and also binds to xcursor with cffi. The bindings are not built at run time and will have to be generated manually when the code is downloaded or when any changes are made to the cffi library. This can be done by calling:

```
./scripts/ffibuild
```

3.2.3 Development and testing

In practice, the development cycle looks something like this:

1. make minor code change
2. run appropriate test: `pytest tests/test_module.py` or `pytest -k PATTERN`
3. GOTO 1, until hackage is complete
4. run entire test suite: `pytest`
5. commit

Of course, your patches should also pass the unit tests as well (i.e. `make check`). These will be run by travis-ci on every pull request so you can see whether or not your contribution passes.

3.2.4 Coding style

While not all of our code follows PEP8, we do try to adhere to it where possible. All new code should be PEP8 compliant.

The `make lint` command will run a linter with our configuration over libqtile to ensure your patch complies with reasonable formatting constraints. We also request that git commit messages follow the standard format.

3.2.5 Deprecation policy

When a widget API is changed, you should deprecate the change using `libqtile.widget.base.deprecated` to warn users, in addition to adding it to the appropriate place in the changelog. We will typically remove deprecated APIs one tag after they are deprecated.

3.2.6 Using Xephyr

Qtile has a very extensive test suite, using the Xephyr nested X server. When tests are run, a nested X server with a nested instance of Qtile is fired up, and then tests interact with the Qtile instance through the client API. The fact that we can do this is a great demonstration of just how completely scriptable Qtile is. In fact, Qtile is designed expressly to be scriptable enough to allow unit testing in a nested environment.

The Qtile repo includes a tiny helper script to let you quickly pull up a nested instance of Qtile in Xephyr, using your current configuration. Run it from the top-level of the repository, like this:

```
./scripts/xephyr
```

Change the screen size by setting the `SCREEN_SIZE` environment variable. Default: 800x600. Example:

```
SCREEN_SIZE=1920x1080 ./scripts/xephyr
```

Change the log level by setting the `LOG_LEVEL` environment variable. Default: INFO. Example:

```
LOG_LEVEL=DEBUG ./scripts/xephyr
```

The script will also pass any additional options to Qtile. For example, you can use a specific configuration file like this:

```
./scripts/xephyr -c ~/.config/qtile/other_config.py
```

Once the Xephyr window is running and focused, you can enable capturing the keyboard shortcuts by hitting Control+Shift. Hitting them again will disable the capture and let you use your personal keyboard shortcuts again.

You can close the Xephyr window by enabling the capture of keyboard shortcuts and hit Mod4+Control+Q. Mod4 (or Mod) is usually the Super key (or Windows key). You can also close the Xephyr window by running `qtile-cmd -o cmd -f shutdown` in a terminal (from inside the Xephyr window of course).

You don't need to run the Xephyr script in order to run the tests as the test runner will launch its own Xephyr instances.

3.2.7 Second X Session

Some users prefer to test Qtile in a second, completely separate X session: Just switch to a new tty and run `startx` normally to use the `~/.xinitrc` X startup script.

It's likely though that you want to use a different, customized startup script for testing purposes, for example `~/.config/qtile/xinitrc`. You can do so by launching X with:

```
startx ~/.config/qtile/xinitrc
```

`startx` deals with multiple X sessions automatically. If you want to use `xinit` instead, you need to first copy `/etc/X11/xinit/xserverrc` to `~/.xserverrc`; when launching it, you have to specify a new session number:

```
xinit ~/.config/qtile/xinitrc -- :1
```

Examples of custom X startup scripts are available in qtile-examples.

3.2.8 Capturing an `xtrace`

Occasionally, a bug will be low level enough to require an `xtrace` of Qtile's conversations with the X server. To capture one of these, create an `xinitrc` or similar file with:

```
exec xtrace qtile >> ~/qtile.log
```

This will put the xtrace output in Qtile's logfile as well. You can then demonstrate the bug, and paste the contents of this file into the bug report.

Note that xtrace may be named `x11trace` on some platforms, for example, on Fedora.

3.2.9 Resources

Here are a number of resources that may come in handy:

- Inter-Client Conventions Manual
- Extended Window Manager Hints
- A reasonable basic Xlib Manual

3.2.10 Troubleshoot

Cairo errors

When running the Xephyr script (`./scripts/xephyr`), you might see tracebacks with attribute errors like the following or similar:

```
AttributeError: cffi library 'libcairo.so.2' has no function, constant or global␣
↪variable named 'cairo_xcb_surface_create'
```

If it happens, it might be because the `cairocffi` and `xcffib` dependencies were installed in the wrong order.

To fix this:

1. uninstall them from your environment: with `pip uninstall cairocffi xcffib` if using a virtualenv, or with your system package-manager if you installed the development version of Qtile system-wide.
2. re-install them sequentially (again, with pip or with your package-manager):

```
pip install xcffib
pip install --no-cache-dir cairocffi
```

See this issue comment for more information.

If you are using your system package-manager and the issue still happens, the packaging of `cairocffi` might be broken for your distribution. Try to contact the persons responsible for `cairocffi`'s packaging on your distribution, or to install it from the sources with `xcffib` available.

DBus/GObject errors

When running the Xephyr script (`./scripts/xephyr`), you might see a line in the output like the following or similar:

```
libqtile manager.py:setup_python_dbus():L310  importing dbus/gobject failed, dbus␣
↪will not work.
```

If it happens, it might be because you are missing some dependencies on your system and/or in your Qtile virtualenv.

To fix this:

1. follow the installation instructions of `PyGObject` at https://pygobject.readthedocs.io/en/latest/getting_started.html. There are methods for several Linux distributions: pick yours.
2. there are instructions for system-wide installation and virtualenv installation: pick the relevant one, depending on how you installed the development version of QTile (usually in a virtualenv).
3. Optionally re-install QTile's dependencies:

```
pip install -r requirements.txt
pip install -r requirements-dev.txt
```

Fonts errors

When running the test suite or the Xephyr script (`./scripts/xephyr`), you might see errors in the output like the following or similar:

- Xephyr script:

```
xterm: cannot load font "-Misc-Fixed-medium-R-*-*-13-120-75-75-C-120-ISO10646-1"
xterm: cannot load font "-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-
↪iso10646-1"
```

- `pytest`:

```
---------- Captured stderr call ----------
Warning: Cannot convert string "8x13" to type FontStruct
Warning: Unable to load any usable ISO8859 font
Warning: Unable to load any usable ISO8859 font
Error: Aborting: no font found

-------- Captured stderr teardown --------
Qtile exited with exitcode: -9
```

If it happens, it might be because you're missing fonts on your system.

On ArchLinux, you can fix this by installing `xorg-fonts-misc`:

```
sudo pacman -S xorg-fonts-misc
```

Try to search for "xorg fonts misc" with your distribution name on the internet to find how to install them.

CHAPTER 4

Miscellaneous

4.1 Reference

4.1.1 Scripting Commands

Here is documented some of the commands available on objects in the command tree when running qshell or scripting commands to qtile. Note that this is an incomplete list, some objects, such as *layouts* and *widgets*, may implement their own set of commands beyond those given here.

Qtile

class libqtile.core.manager.**Qtile**(*kore*, *config*, *eventloop*, *no_spawn=False*, *state=None*)

This object is the *root* of the command graph

cmd_add_rule(*self*, *match_args*, *rule_args*, *min_priorty=False*)

Add a dgroup rule, returns rule_id needed to remove it

Parameters

match_args : config.Match arguments

rule_args : config.Rule arguments

min_priorty : If the rule is added with minimum prioriry (last) (default: False)

cmd_addgroup(*self*, *group*, *label=None*, *layout=None*, *layouts=None*)

Add a group with the given name

cmd_commands(*self*) → List[str]

Returns a list of possible commands for this object

Used by __qsh__ for command completion and online help

cmd_critical(*self*)

Set log level to CRITICAL

cmd_debug(*self*)
Set log level to DEBUG

cmd_delgroup(*self*, *group*)
Delete a group with the given name

cmd_display_kb(*self*, **args*)
Display table of key bindings

cmd_doc(*self*, *name*) → str
Returns the documentation for a specified command name

Used by __qsh__ to provide online help.

cmd_error(*self*)
Set log level to ERROR

cmd_eval(*self*, *code: str*) → Tuple[bool, Union[str, NoneType]]
Evaluates code in the same context as this function

Return value is tuple *(success, result)*, success being a boolean and result being a string representing the return value of eval, or None if exec was used instead.

cmd_findwindow(*self*, *prompt='window'*, *widget='prompt'*)
Launch prompt widget to find a window of the given name

Parameters

prompt : Text with which to prompt user (default: "window")

widget : Name of the prompt widget (default: "prompt")

cmd_focus_by_click(*self*, *e*)
Bring a window to the front

Parameters

e [xcb event] Click event used to determine window to focus

cmd_function(*self*, *function*, **args*, ***kwargs*) → None
Call a function with current object as argument

cmd_get_info(*self*)
Prints info for all groups

cmd_get_state(*self*)
Get pickled state for restarting qtile

cmd_get_test_data(*self*)
Returns any content arbitrarily set in the self.test_data attribute. Useful in tests.

cmd_groups(*self*)
Return a dictionary containing information for all groups

Examples

groups()

cmd_hide_show_bar(*self*, *position='all'*)
Toggle visibility of a given bar

Parameters

position : one of: "top", "bottom", "left", "right", or "all" (default: "all")

cmd_info(*self*)
Set log level to INFO

cmd_internal_windows(*self*)
Return info for each internal window (bars, for example)

cmd_items(*self*, *name*) → Tuple[bool, List[str]]
Returns a list of contained items for the specified name

Used by __qsh__ to allow navigation of the object graph.

cmd_list_widgets(*self*)
List of all addressible widget names

cmd_loglevel(*self*)

cmd_loglevelname(*self*)

cmd_next_layout(*self*, *group=None*)
Switch to the next layout.

Parameters

group : Group name. If not specified, the current group is assumed

cmd_next_screen(*self*)
Move to next screen

cmd_next_urgent(*self*)
Focus next window with urgent hint

cmd_pause(*self*)
Drops into pdb

cmd_prev_layout(*self*, *group=None*)
Switch to the previous layout.

Parameters

group : Group name. If not specified, the current group is assumed

cmd_prev_screen(*self*)
Move to the previous screen

cmd_qtile_info(*self*)
Returns a dictionary of info on the Qtile instance

cmd_qtilecmd(*self*, *prompt='command'*, *widget='prompt'*, *messenger='xmessage'*) → None
Execute a Qtile command using the client syntax

Tab completion aids navigation of the command tree

Parameters

prompt : Text to display at the prompt (default: "command: ")

widget : Name of the prompt widget (default: "prompt")

messenger : Command to display output, set this to None to disable (default: "xmessage")

cmd_remove_rule(*self*, *rule_id*)
Remove a dgroup rule by rule_id

cmd_restart(*self*)
Restart qtile

cmd_run_extension(*self*, *extension*)
Run extensions

cmd_run_external(*self*, *full_path*)
Run external Python script

cmd_screens(*self*)
Return a list of dictionaries providing information on all screens

cmd_shutdown(*self*)
Quit Qtile

cmd_simulate_keypress(*self*, *modifiers*, *key*)
Simulates a keypress on the focused window.

Parameters

modifiers : A list of modifier specification strings. Modifiers can be one of "shift", "lock", "control" and "mod1" - "mod5".

key : Key specification.

Examples

simulate_keypress(["control", "mod2"], "k")

cmd_spawn(*self*, *cmd*)
Run cmd in a shell.

cmd may be a string, which is parsed by shlex.split, or a list (similar to subprocess.Popen).

Examples

spawn("firefox")

spawn(["xterm", "-T", "Temporary terminal"])

cmd_spawncmd(*self*, *prompt='spawn'*, *widget='prompt'*, *command='%s'*, *complete='cmd'*)
Spawn a command using a prompt widget, with tab-completion.

Parameters

prompt : Text with which to prompt user (default: "spawn: ").

widget : Name of the prompt widget (default: "prompt").

command : command template (default: "%s").

complete : Tab completion function (default: "cmd")

cmd_status(*self*)
Return "OK" if Qtile is running

cmd_switch_groups(*self*, *groupa*, *groupb*)
Switch position of groupa to groupb

cmd_switchgroup(*self*, *prompt='group'*, *widget='prompt'*)
Launch prompt widget to switch to a given group to the current screen

Parameters

prompt : Text with which to prompt user (default: "group")

widget : Name of the prompt widget (default: "prompt")

cmd_sync(*self*)
Sync the X display. Should only be used for development

cmd_to_layout_index(*self*, *index*, *group=None*)
Switch to the layout with the given index in self.layouts.

Parameters

index : Index of the layout in the list of layouts.

group : Group name. If not specified, the current group is assumed.

cmd_to_screen(*self*, *n*)
Warp focus to screen n, where n is a 0-based screen number

Examples

to_screen(0)

cmd_togroup(*self*, *prompt='group'*, *widget='prompt'*)
Launch prompt widget to move current window to a given group

Parameters

prompt : Text with which to prompt user (default: "group")

widget : Name of the prompt widget (default: "prompt")

cmd_tracemalloc_dump(*self*)
Dump tracemalloc snapshot

cmd_tracemalloc_toggle(*self*)
Toggle tracemalloc status

Running tracemalloc is required for qtile-top

cmd_warning(*self*)
Set log level to WARNING

cmd_windows(*self*)
Return info for each client window

Bar

class libqtile.bar.**Bar**(*widgets*, *size*, ***config*)
A bar, which can contain widgets

Parameters

widgets : A list of widget objects.

size : The "thickness" of the bar, i.e. the height of a horizontal bar, or the width of a vertical bar.

key	default	description
background	'#000000'	Background colour.
opacity	1	Bar window opacity.

cmd_commands(*self*) → List[str]
Returns a list of possible commands for this object

Used by __qsh__ for command completion and online help

cmd_doc(*self*, *name*) → str
Returns the documentation for a specified command name

Used by __qsh__ to provide online help.

cmd_eval(*self*, *code: str*) → Tuple[bool, Union[str, NoneType]]
Evaluates code in the same context as this function

Return value is tuple *(success, result)*, success being a boolean and result being a string representing the return value of eval, or None if exec was used instead.

cmd_fake_button_press(*self*, *screen*, *position*, *x*, *y*, *button=1*)
Fake a mouse-button-press on the bar. Co-ordinates are relative to the top-left corner of the bar.

:screen The integer screen offset :position One of "top", "bottom", "left", or "right"

cmd_function(*self*, *function*, **args*, ***kwargs*) → None
Call a function with current object as argument

cmd_info(*self*)
Info for this object.

cmd_items(*self*, *name*) → Tuple[bool, List[str]]
Returns a list of contained items for the specified name

Used by __qsh__ to allow navigation of the object graph.

Group

class libqtile.config.**Group**(*name*, *matches=None*, *exclusive=False*, *spawn=None*, *layout=None*, *layouts=None*, *persist=True*, *init=True*, *layout_opts=None*, *screen_affinity=None*, *position=9223372036854775807*, *label=None*)
Represents a "dynamic" group

These groups can spawn apps, only allow certain Matched windows to be on them, hide when they're not in use, etc. Groups are identified by their name.

Parameters

name [string] the name of this group

matches [default `None`] list of `Match` objects whose windows will be assigned to this group

exclusive [boolean] when other apps are started in this group, should we allow them here or not?

spawn [string or list of strings] this will be `exec()` d when the group is created, you can pass either a program name or a list of programs to `exec()`

layout [string] the name of default layout for this group (e.g. 'max' or 'stack'). This is the name specified for a particular layout in config.py or if not defined it defaults in general the class name in all lower case.

layouts [list] the group layouts list overriding global layouts. Use this to define a separate list of layouts for this particular group.

persist [boolean] should this group stay alive with no member windows?

init [boolean] is this group alive when qtile starts?

position [int] group position

label [string] the display name of the group. Use this to define a display name other than name of the group. If set to None, the display name is set to the name.

Screen

class libqtile.config.**Screen**(*top=None*, *bottom=None*, *left=None*, *right=None*, *x=None*, *y=None*, *width=None*, *height=None*)

A physical screen, and its associated paraphernalia.

Define a screen with a given set of Bars of a specific geometry. Note that bar.Bar objects can only be placed at the top or the bottom of the screen (bar.Gap objects can be placed anywhere). Also, `x`, `y`, `width`, and `height` aren't specified usually unless you are using 'fake screens'.

Parameters

top: Gap/Bar object, or None.

bottom: Gap/Bar object, or None.

left: Gap/Bar object, or None.

right: Gap/Bar object, or None.

x [int or None]

y [int or None]

width [int or None]

height [int or None]

cmd_commands(*self*) → List[str]

Returns a list of possible commands for this object

Used by __qsh__ for command completion and online help

cmd_doc(*self*, *name*) → str

Returns the documentation for a specified command name

Used by __qsh__ to provide online help.

cmd_eval(*self*, *code: str*) → Tuple[bool, Union[str, NoneType]]

Evaluates code in the same context as this function

Return value is tuple *(success, result)*, success being a boolean and result being a string representing the return value of eval, or None if exec was used instead.

cmd_function(*self*, *function*, **args*, ***kwargs*) → None

Call a function with current object as argument

cmd_info(*self*)

Returns a dictionary of info for this screen.

cmd_items(*self*, *name*) → Tuple[bool, List[str]]

Returns a list of contained items for the specified name

Used by __qsh__ to allow navigation of the object graph.

cmd_next_group(*self*, *skip_empty=False*, *skip_managed=False*)

Switch to the next group

cmd_prev_group(*self*, *skip_empty=False*, *skip_managed=False*)
Switch to the previous group

cmd_resize(*self*, *x=None*, *y=None*, *w=None*, *h=None*)
Resize the screen

cmd_toggle_group(*self*, *group_name=None*)
Switch to the selected group or to the previously active one

cmd_togglegroup(*self*, *groupName=None*)
Switch to the selected group or to the previously active one

Deprecated: use toggle_group()

Window

class libqtile.window.**Window**(*window*, *qtile*)

cmd_bring_to_front(*self*)

cmd_commands(*self*) → List[str]
Returns a list of possible commands for this object

Used by __qsh__ for command completion and online help

cmd_disable_floating(*self*)

cmd_disable_fullscreen(*self*)

cmd_disable_maximize(*self*)

cmd_disable_minimize(*self*)

cmd_doc(*self*, *name*) → str
Returns the documentation for a specified command name

Used by __qsh__ to provide online help.

cmd_down_opacity(*self*)

cmd_enable_floating(*self*)

cmd_enable_fullscreen(*self*)

cmd_enable_maximize(*self*)

cmd_enable_minimize(*self*)

cmd_eval(*self*, *code: str*) → Tuple[bool, Union[str, NoneType]]
Evaluates code in the same context as this function

Return value is tuple *(success, result)*, success being a boolean and result being a string representing the return value of eval, or None if exec was used instead.

cmd_focus(*self*, *warp=None*)
Focuses the window.

cmd_function(*self*, *function*, **args*, ***kwargs*) → None
Call a function with current object as argument

cmd_get_position(*self*)

cmd_get_size(*self*)

cmd_info(*self*)
Returns a dictionary of info for this object

cmd_inspect(*self*)
Tells you more than you ever wanted to know about a window

cmd_items(*self*, *name*) → Tuple[bool, List[str]]
Returns a list of contained items for the specified name

Used by __qsh__ to allow navigation of the object graph.

cmd_kill(*self*)
Kill this window

Try to do this politely if the client support this, otherwise be brutal.

cmd_match(*self*, **args*, ***kwargs*)

cmd_move_floating(*self*, *dx*, *dy*)
Move window by dx and dy

cmd_opacity(*self*, *opacity*)

cmd_resize_floating(*self*, *dw*, *dh*)
Add dw and dh to size of window

cmd_set_position(*self*, *dx*, *dy*)

cmd_set_position_floating(*self*, *x*, *y*)
Move window to x and y

cmd_set_size_floating(*self*, *w*, *h*)
Set window dimensions to w and h

cmd_static(*self*, *screen*, *x*, *y*, *width*, *height*)

cmd_toggle_floating(*self*)

cmd_toggle_fullscreen(*self*)

cmd_toggle_maximize(*self*)

cmd_toggle_minimize(*self*)

cmd_togroup(*self*, *groupName=None*, ***, *switch_group=False*)
Move window to a specified group.

If groupName is not specified, we assume the current group. If switch_group is True, also switch to that group.

Examples

Move window to current group:

```
togroup()
```

Move window to group “a”:

```
togroup("a")
```

Move window to group “a”, and switch to group “a”:

```
togroup("a", switch_group=True)
```

cmd_toscreen(*self*, *index=None*)
Move window to a specified screen.

If index is not specified, we assume the current screen

Examples

Move window to current screen:

```
toscreen()
```

Move window to screen 0:

```
toscreen(0)
```

cmd_up_opacity(*self*)

4.1.2 Built-in Hooks

classmethod subscribe.**addgroup**(*func*)
Called when group is added

Arguments

- qtile manager instance
- name of new group

classmethod subscribe.**changegroup**(*func*)
Called whenever a group change occurs

Arguments

None

classmethod subscribe.**client_focus**(*func*)
Called whenever focus changes

Arguments

- window.Window object of the new focus.

classmethod subscribe.**client_killed**(*func*)
Called after a client has been unmanaged

Arguments

- window.Window object of the killed window.

classmethod subscribe.**client_managed**(*func*)
Called after Qtile starts managing a new client

Called after a window is assigned to a group, or when a window is made static. This hook is not called for internal windows.

Arguments

- window.Window object of the managed window

classmethod subscribe.**client_mouse_enter**(*func*)
Called when the mouse enters a client

Arguments

- window.Window of window entered

classmethod subscribe.**client_name_updated**(*func*)
Called when the client name changes

Arguments

- window.Window of client with updated name

classmethod subscribe.**client_new**(*func*)
Called before Qtile starts managing a new client

Use this hook to declare windows static, or add them to a group on startup. This hook is not called for internal windows.

Arguments

- window.Window object

Examples

```
@libqtile.hook.subscribe.client_new
def func(c):
    if c.name == "xterm":
        c.togroup("a")
    elif c.name == "dzen":
        c.static(0)
```

classmethod subscribe.**client_state_changed**(*func*)
Called whenever client state changes

Never fires

classmethod subscribe.**client_type_changed**(*func*)
Called whenever window type changes

Never fires

classmethod subscribe.**client_urgent_hint_changed**(*func*)
Called when the client urgent hint changes

Arguments

- window.Window of client with hint change

classmethod subscribe.**current_screen_change**(*func*)
Called when the current screen (i.e. the screen with focus) changes

Arguments

None

classmethod subscribe.**delgroup**(*func*)
Called when group is deleted

Arguments

- qtile manager instance

- name of deleted group

classmethod subscribe.**float_change**(*func*)
Called when a change in float state is made

Arguments

None

classmethod subscribe.**focus_change**(*func*)
Called when focus is changed

Arguments

None

classmethod subscribe.**group_window_add**(*func*)
Called when a new window is added to a group

Arguments

None

classmethod subscribe.**layout_change**(*func*)
Called on layout change

Arguments

- layout object for new layout
- group object on which layout is changed

classmethod subscribe.**net_wm_icon_change**(*func*)
Called on *_NET_WM_ICON* chance

Arguments

- `window.Window` of client with changed icon

classmethod subscribe.**screen_change**(*func*)
Called when a screen is added or screen configuration is changed (via xrandr)

Common usage is simply to call `qtile.cmd_restart()` on each event (to restart qtile when there is a new monitor):

Arguments

- qtile manager instance
- `xproto.randr.ScreenChangeNotify` event

Examples

```
@libqtile.hook.subscribe.screen_change
def restart_on_randr(qtile, ev):
    qtile.cmd_restart()
```

classmethod subscribe.**selection_change**(*func*)
Called on selection change

Arguments

- name of the selection
- dictionary describing selection, containing `owner` and `selection` as keys

classmethod subscribe.**selection_notify**(*func*)
Called on selection notify

Arguments

- name of the selection
- dictionary describing selection, containing `owner` and `selection` as keys

classmethod subscribe.**setgroup**(*func*)
Called when group is changed

Arguments

None

classmethod subscribe.**startup**(*func*)
Called when qtile is started

Arguments

None

classmethod subscribe.**startup_complete**(*func*)
Called when qtile is started after all resources initialized

Arguments

None

classmethod subscribe.**startup_once**(*func*)
Called when Qtile has started on first start

This hook is called exactly once per session (i.e. not on each `lazy.restart()`).

Arguments

None

classmethod subscribe.**window_name_change**(*func*)
Called whenever a windows name changes

Deprecated: use client_name_updated **Arguments**

None

4.1.3 Built-in Layouts

Floating

class libqtile.layout.floating.**Floating**(*float_rules=None*, *no_reposition_match=None*, ***config*)
Floating layout, which does nothing with windows but handles focus order

key	default	description
auto_float_types	{'notification', 'splash', 'dialog', 'utility', 'toolbar'}	default wm types to automatically float
border_focus	'#0000ff'	Border colour for the focused window.
border_normal	'#000000'	Border colour for un-focused windows.
border_width	1	Border width.
fullscreen_border_width	0	Border width for fullscreen.
max_border_width	0	Border width for maximize.
name	'floating'	Name of this layout.

Bsp

class libqtile.layout.bsp.**Bsp**(***config*)

This layout is inspired by bspwm, but it does not try to copy its features.

The first client occupies the entire srceen space. When a new client is created, the selected space is partitioned in 2 and the new client occupies one of those subspaces, leaving the old client with the other.

The partition can be either horizontal or vertical according to the dimensions of the current space: if its width/height ratio is above a pre-configured value, the subspaces are created side-by-side, otherwise, they are created on top of each other. The partition direction can be freely toggled. All subspaces can be resized and clients can be shuffled around.

An example key configuration is:

```
Key([mod], "j", lazy.layout.down()),
Key([mod], "k", lazy.layout.up()),
Key([mod], "h", lazy.layout.left()),
Key([mod], "l", lazy.layout.right()),
Key([mod, "shift"], "j", lazy.layout.shuffle_down()),
Key([mod, "shift"], "k", lazy.layout.shuffle_up()),
Key([mod, "shift"], "h", lazy.layout.shuffle_left()),
Key([mod, "shift"], "l", lazy.layout.shuffle_right()),
Key([mod, "mod1"], "j", lazy.layout.flip_down()),
Key([mod, "mod1"], "k", lazy.layout.flip_up()),
Key([mod, "mod1"], "h", lazy.layout.flip_left()),
Key([mod, "mod1"], "l", lazy.layout.flip_right()),
Key([mod, "control"], "j", lazy.layout.grow_down()),
Key([mod, "control"], "k", lazy.layout.grow_up()),
Key([mod, "control"], "h", lazy.layout.grow_left()),
Key([mod, "control"], "l", lazy.layout.grow_right()),
Key([mod, "shift"], "n", lazy.layout.normalize()),
Key([mod], "Return", lazy.layout.toggle_split()),
```

key	default	description
border_focus	'#881111'	Border colour for the focused window.
border_normal	'#220000'	Border colour for un-focused windows.
border_width	2	Border width.
fair	True	New clients are inserted in the shortest branch.
grow_amount	10	Amount by which to grow a window/column.
lower_right	True	New client occupies lower or right subspace.
margin	0	Margin of the layout.
name	'bsp'	Name of this layout.
ratio	1.6	Width/height ratio that defines the partition direction.

Columns

class libqtile.layout.columns.**Columns**(***config*)

Extension of the Stack layout.

The screen is split into columns, which can be dynamically added or removed. Each column can present its windows in 2 modes: split or stacked. In split mode, all windows are presented simultaneously, spliting the column space. In stacked mode, only a single window is presented from the stack of windows. Columns and windows can be resized and windows can be shuffled around.

This layout can also emulate wmii's default layout via:

layout.Columns(num_columns=1, insert_position=1)

Or the "Vertical", and "Max", depending on the default parameters.

An example key configuration is:

```
Key([mod], "j", lazy.layout.down()),
Key([mod], "k", lazy.layout.up()),
Key([mod], "h", lazy.layout.left()),
Key([mod], "l", lazy.layout.right()),
Key([mod, "shift"], "j", lazy.layout.shuffle_down()),
Key([mod, "shift"], "k", lazy.layout.shuffle_up()),
Key([mod, "shift"], "h", lazy.layout.shuffle_left()),
Key([mod, "shift"], "l", lazy.layout.shuffle_right()),
Key([mod, "control"], "j", lazy.layout.grow_down()),
Key([mod, "control"], "k", lazy.layout.grow_up()),
Key([mod, "control"], "h", lazy.layout.grow_left()),
Key([mod, "control"], "l", lazy.layout.grow_right()),
Key([mod], "Return", lazy.layout.toggle_split()),
Key([mod], "n", lazy.layout.normalize()),
```

key	default	description
border_focus	'#881111'	Border colour for the focused window.
border_focus_stack	'#881111'	Border colour for the focused window in stacked columns.
border_normal	'#220000'	Border colour for un-focused windows.
border_normal_stack	'#220000'	Border colour for un-focused windows in stacked columns.
border_width	2	Border width.
fair	False	Add new windows to the column with least windows.
grow_amount	10	Amount by which to grow a window/column.
insert_position	0	Position relative to the current window where new ones are inserted (0 means right above the current window, 1 means right after).
margin	0	Margin of the layout.
name	'columns'	Name of this layout.
num_columns	2	Preferred number of columns.
split	True	New columns presentation mode.
wrap_focus_columns	True	Wrap the screen when moving focus across columns.
wrap_focus_rows	True	Wrap the screen when moving focus across rows.
wrap_focus_stack	True	Wrap the screen when moving focus across stacked.

Matrix

class libqtile.layout.matrix.**Matrix**(*columns=2*, ***config*)

This layout divides the screen into a matrix of equally sized cells and places one window in each cell. The number of columns is configurable and can also be changed interactively.

key	default	description
border_focus	'#0000ff'	Border colour for the focused window.
border_normal	'#000000'	Border colour for un-focused windows.
border_width	1	Border width.
margin	0	Margin of the layout
name	'matrix'	Name of this layout.

Max

class libqtile.layout.max.**Max**(***config*)

Maximized layout

A simple layout that only displays one window at a time, filling the screen. This is suitable for use on laptops and other devices with small screens. Conceptually, the windows are managed as a stack, with commands to switch to next and previous windows in the stack.

key	default	description
name	'max'	Name of this layout.

MonadTall

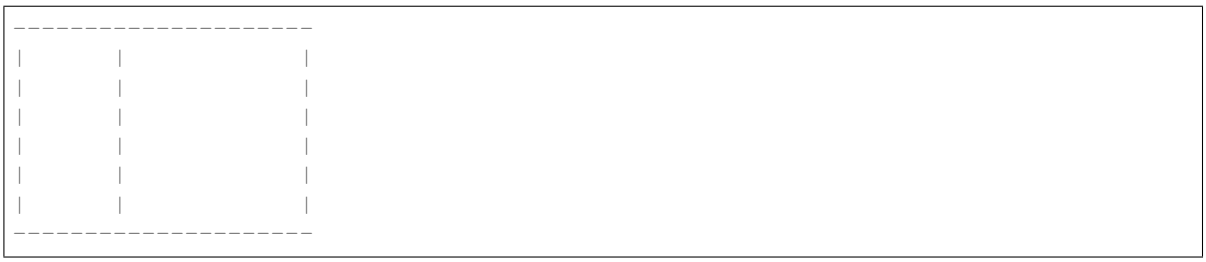
class libqtile.layout.xmonad.**MonadTall**(***config*)

Emulate the behavior of XMonad's default tiling scheme.

Main-Pane:

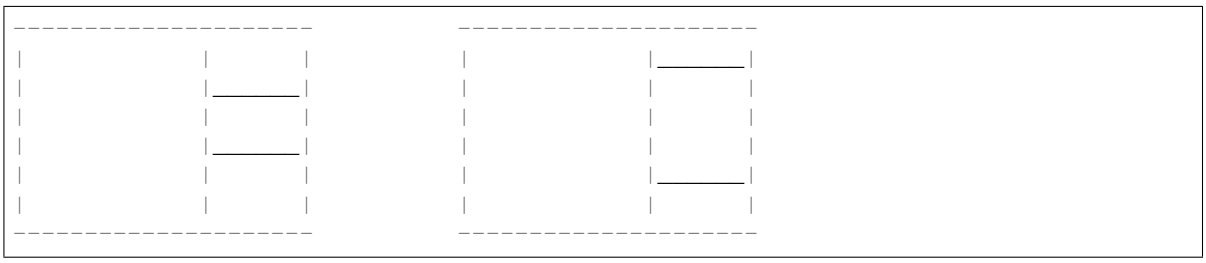
A main pane that contains a single window takes up a vertical portion of the screen based on the ratio setting. This ratio can be adjusted with the `cmd_grow_main` and `cmd_shrink_main` or, while the main pane is in focus, `cmd_grow` and `cmd_shrink`.

Using the `cmd_flip` method will switch which horizontal side the main pane will occupy. The main pane is considered the "top" of the stack.



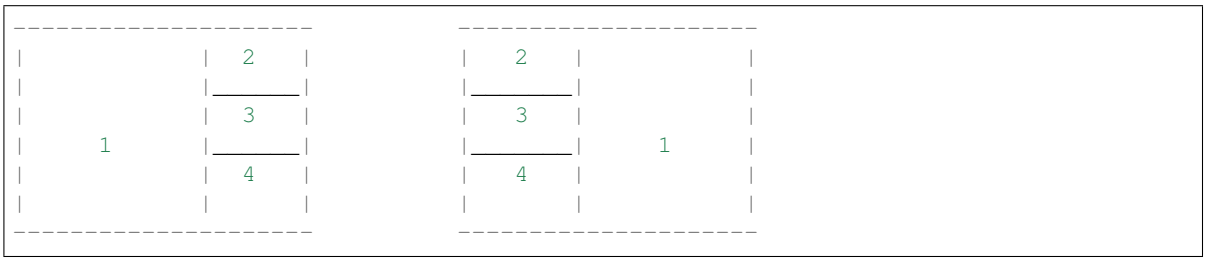
Secondary-panes:

Occupying the rest of the screen are one or more secondary panes. The secondary panes will share the vertical space of the screen however they can be resized at will with the `cmd_grow` and `cmd_shrink` methods. The other secondary panes will adjust their sizes to smoothly fill all of the space.



Panes can be moved with the `cmd_shuffle_up` and `cmd_shuffle_down` methods. As mentioned the main pane is considered the top of the stack; moving up is counter-clockwise and moving down is clockwise.

The opposite is true if the layout is "flipped".



Normalizing:

To restore all client windows to their default size ratios simply use the `cmd_normalize` method.

Maximizing:

To toggle a client window between its minimum and maximum sizes simply use the `cmd_maximize` on a focused client.

Suggested Bindings:

```
Key([modkey], "h", lazy.layout.left()),
Key([modkey], "l", lazy.layout.right()),
Key([modkey], "j", lazy.layout.down()),
Key([modkey], "k", lazy.layout.up()),
Key([modkey, "shift"], "h", lazy.layout.swap_left()),
Key([modkey, "shift"], "l", lazy.layout.swap_right()),
Key([modkey, "shift"], "j", lazy.layout.shuffle_down()),
Key([modkey, "shift"], "k", lazy.layout.shuffle_up()),
Key([modkey], "i", lazy.layout.grow()),
Key([modkey], "m", lazy.layout.shrink()),
Key([modkey], "n", lazy.layout.normalize()),
Key([modkey], "o", lazy.layout.maximize()),
Key([modkey, "shift"], "space", lazy.layout.flip()),
```

key	default	description
`align`	`0`	Which side master plane will be placed (one of `MonadTall._left` or `MonadTall._right`)
`border_focus`	`'#ff0000'`	Border colour for the focused window.
`border_normal`	`'#000000'`	Border colour for un-focused windows.
`border_width`	`2`	Border width.
`change_ratio`	`0.05`	Resize ratio
`change_size`	`20`	Resize change in pixels
`margin`	`0`	Margin of the layout
`max_ratio`	`0.75`	The percent of the screen-space the master pane should occupy at maximum.
`min_ratio`	`0.25`	The percent of the screen-space the master pane should occupy at minimum.
`min_secondary_size`	`85`	minimum size in pixel for a secondary pane window
`name`	`'xmonadtall'`	Name of this layout.
`new_at_current`	`False`	Place new windows at the position of the active window.
`ratio`	`0.5`	The percent of the screen-space the master pane should occupy by default.
`single_border_width`	`None`	Border width for single window
`single_margin`	`None`	Margin size for single window

MonadWide

class libqtile.layout.xmonad.**MonadWide**(***config*)

Emulate the behavior of XMonad's horizontal tiling scheme.

This layout attempts to emulate the behavior of XMonad wide tiling scheme.

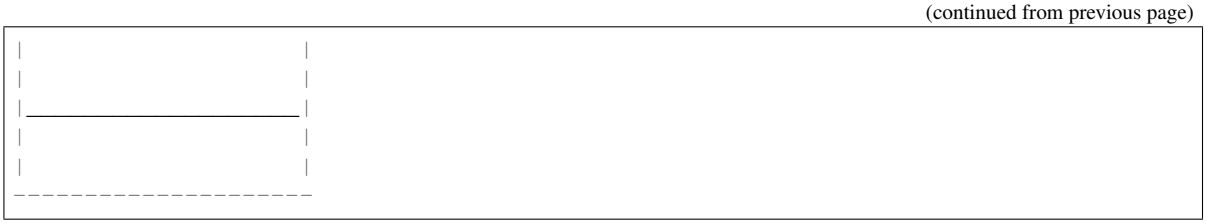
Main-Pane:

A main pane that contains a single window takes up a horizontal portion of the screen based on the ratio setting. This ratio can be adjusted with the `cmd_grow_main` and `cmd_shrink_main` or, while the main pane is in focus, `cmd_grow` and `cmd_shrink`.

```
----------------------
|                    |
```

(continues on next page)

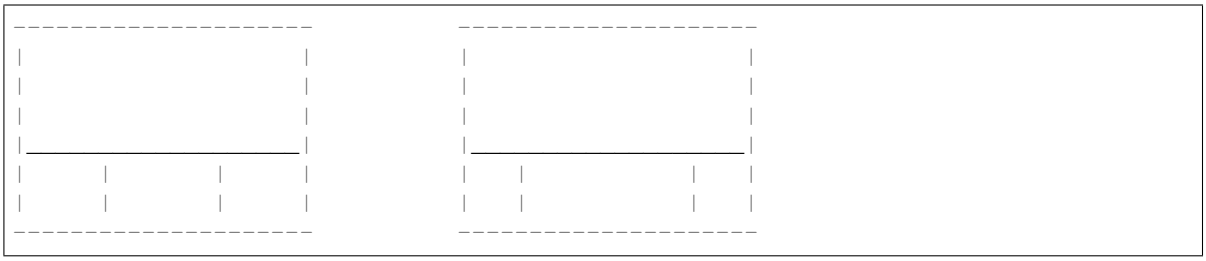
(continued from previous page)



Using the `cmd_flip` method will switch which vertical side the main pane will occupy. The main pane is considered the "top" of the stack.

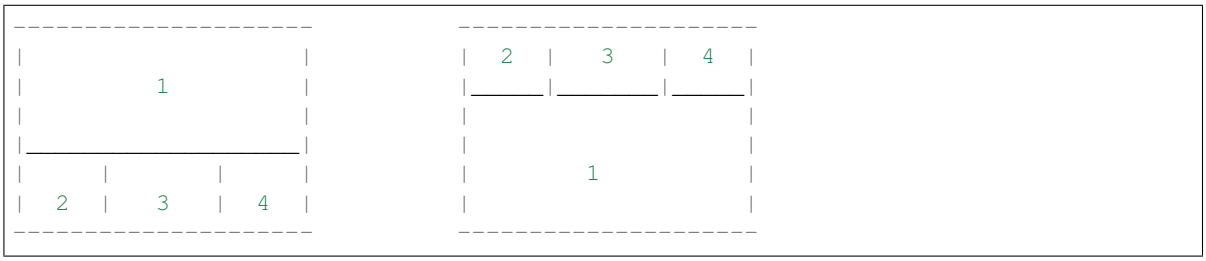
Secondary-panes:

Occupying the rest of the screen are one or more secondary panes. The secondary panes will share the horizontal space of the screen however they can be resized at will with the `cmd_grow` and `cmd_shrink` methods. The other secondary panes will adjust their sizes to smoothly fill all of the space.



Panes can be moved with the `cmd_shuffle_up` and `cmd_shuffle_down` methods. As mentioned the main pane is considered the top of the stack; moving up is counter-clockwise and moving down is clockwise.

The opposite is true if the layout is "flipped".



Normalizing:

To restore all client windows to their default size ratios simply use the `cmd_normalize` method.

Maximizing:

To toggle a client window between its minimum and maximum sizes simply use the `cmd_maximize` on a focused client.

Suggested Bindings:

```
Key([modkey], "h", lazy.layout.left()),
Key([modkey], "l", lazy.layout.right()),
Key([modkey], "j", lazy.layout.down()),
Key([modkey], "k", lazy.layout.up()),
Key([modkey, "shift"], "h", lazy.layout.swap_left()),
Key([modkey, "shift"], "l", lazy.layout.swap_right()),
Key([modkey, "shift"], "j", lazy.layout.shuffle_down()),
Key([modkey, "shift"], "k", lazy.layout.shuffle_up()),
Key([modkey], "i", lazy.layout.grow()),
Key([modkey], "m", lazy.layout.shrink()),
Key([modkey], "n", lazy.layout.normalize()),
Key([modkey], "o", lazy.layout.maximize()),
Key([modkey, "shift"], "space", lazy.layout.flip()),
```

key	default	description
align	0	Which side master plane will be placed (one of MonadTall._left or MonadTall._right)
border_focus	'#ff0000'	Border colour for the focused window.
border_normal	'#000000'	Border colour for un-focused windows.
border_width	2	Border width.
change_ratio	0.05	Resize ratio
change_size	20	Resize change in pixels
margin	0	Margin of the layout
max_ratio	0.75	The percent of the screen-space the master pane should occupy at maximum.
min_ratio	0.25	The percent of the screen-space the master pane should occupy at minimum.
min_secondary_size	85	minimum size in pixel for a secondary pane window
name	'xmonadtall'	Name of this layout.
new_at_current	False	Place new windows at the position of the active window.
ratio	0.5	The percent of the screen-space the master pane should occupy by default.
single_border_width	None	Border width for single window
single_margin	None	Margin size for single window

RatioTile

class libqtile.layout.ratiotile.**RatioTile**(***config*)

Tries to tile all windows in the width/height ratio passed in

key	default	description
border_focus	'#0000ff'	Border colour for the focused window.
border_normal	'#000000'	Border colour for un-focused windows.
border_width	1	Border width.
fancy	False	Use a different method to calculate window sizes.
margin	0	Margin of the layout
name	'ratiotile'	Name of this layout.
ratio	1.618	Ratio of the tiles
ratio_increment	0.1	Amount to increment per ratio increment

Slice

class libqtile.layout.slice.**Slice**(***config*)

Slice layout

This layout cuts piece of screen and places a single window on that piece, and delegates other window placement to other layout

key	default	description
fallback	<libqtile.layout.max.Max object at 0x7fa62cfcdf28>	Fallback layout
name	'max'	Name of this layout.
role	None	WM_WINDOW_ROLE to match
side	'left'	Side of the slice (left, right, top, bottom)
width	256	Slice width
wmclass	None	WM_CLASS to match
wname	None	WM_NAME to match

Stack

class libqtile.layout.stack.**Stack**(***config*)

A layout composed of stacks of windows

The stack layout divides the screen horizontally into a set of stacks. Commands allow you to switch between stacks, to next and previous windows within a stack, and to split a stack to show all windows in the stack, or unsplit it to show only the current window.

Unlike the columns layout the number of stacks is fixed.

key	default	description
autosplit	False	Auto split all new stacks.
border_focus	'#0000ff'	Border colour for the focused window.
border_normal	'#000000'	Border colour for un-focused windows.
border_width	1	Border width.
fair	False	Add new windows to the stacks in a round robin way.
margin	0	Margin of the layout
name	'stack'	Name of this layout.
num_stacks	2	Number of stacks.

Tile

class libqtile.layout.tile.**Tile**(*ratio=0.618*, *masterWindows=1*, *expand=True*, *ratio_increment=0.05*, *add_on_top=True*, *add_after_last=False*, *shift_windows=False*, *master_match=None*, ***config*)

key	default	description
border_focus	'#0000ff'	Border colour for the focused window.
border_normal	'#000000'	Border colour for un-focused windows.
border_width	1	Border width.
margin	0	Margin of the layout
name	'tile'	Name of this layout.

TreeTab

class libqtile.layout.tree.**TreeTab**(***config*)

Tree Tab Layout

This layout works just like Max but displays tree of the windows at the left border of the screen, which allows you to overview all opened windows. It's designed to work with `uzbl-browser` but works with other windows too.

key	default	description
active_bg	'000080'	Background color of active tab
active_fg	'ffffff'	Foreground color of active tab
bg_color	'000000'	Background color of tabs
border_width	2	Width of the border
font	'sans'	Font
fontshadow	None	font shadow color, default is None (no shadow)
fontsize	14	Font pixel size.
inactive_bg	'606060'	Background color of inactive tab
inactive_fg	'ffffff'	Foreground color of inactive tab
level_shift	8	Shift for children tabs
margin_left	6	Left margin of tab panel
margin_y	6	Vertical margin of tab panel
name	'treetab'	Name of this layout.
padding_left	6	Left padding for tabs
padding_x	6	Left padding for tab label
padding_y	2	Top padding for tab label
panel_width	150	Width of the left panel
previous_on_rm	False	Focus previous window on close instead of first.
section_bottom	6	Bottom margin of section
section_fg	'ffffff'	Color of section label
section_fontsize	11	Font pixel size of section label
section_left	4	Left margin of section label
section_padding	4	Bottom of margin section label
section_top	4	Top margin of section label
sections	['Default']	Foreground color of inactive tab
vspace	2	Space between tabs

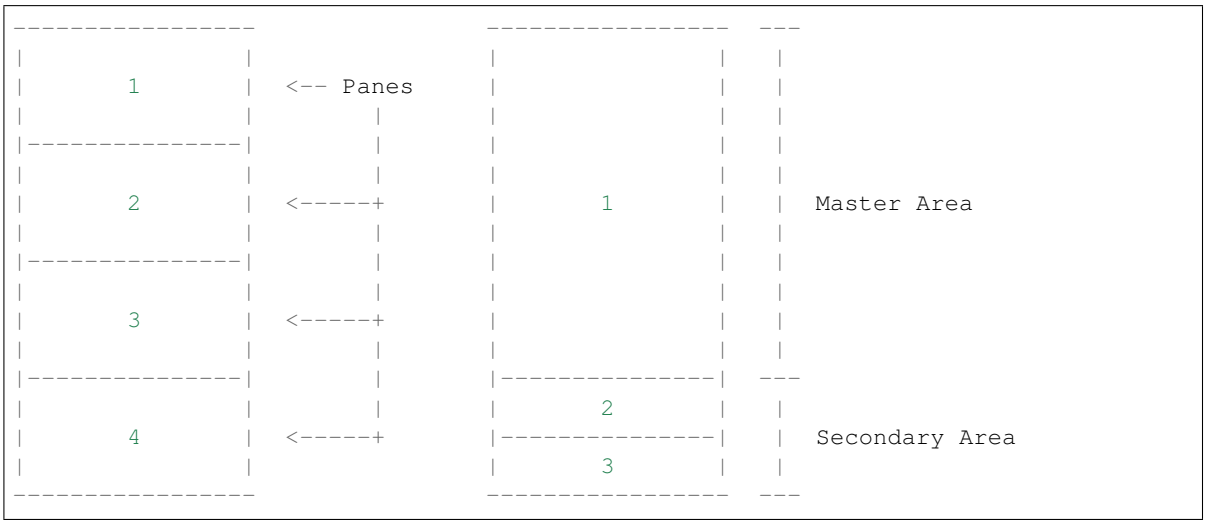
VerticalTile

class libqtile.layout.verticaltile.**VerticalTile**(***config*)

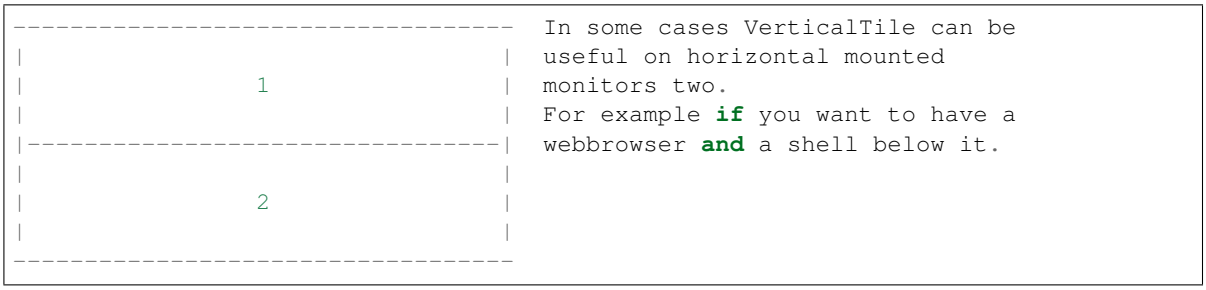
Tiling layout that works nice on vertically mounted monitors

The available height gets divided by the number of panes, if no pane is maximized. If one pane has been maximized, the available height gets split in master- and secondary area. The maximized pane (master pane)

gets the full height of the master area and the other panes (secondary panes) share the remaining space. The master area (at default 75%) can grow and shrink via keybindings.



Normal behavior. No One maximized pane in the master area maximized pane. No and two secondary panes in the specific areas. secondary area.



Suggested keybindings:

```
Key([modkey], 'j', lazy.layout.down()),
Key([modkey], 'k', lazy.layout.up()),
Key([modkey], 'Tab', lazy.layout.next()),
Key([modkey, 'shift'], 'Tab', lazy.layout.next()),
Key([modkey, 'shift'], 'j', lazy.layout.shuffle_down()),
Key([modkey, 'shift'], 'k', lazy.layout.shuffle_up()),
Key([modkey], 'm', lazy.layout.maximize()),
Key([modkey], 'n', lazy.layout.normalize()),
```

key	default	description
border_focus	'#FF0000'	Border color for the focused window.
border_normal	'#FFFFFF'	Border color for un-focused windows.
border_width	1	Border width.
margin	0	Border margin.
name	'verticaltile'	Name of this layout.

Zoomy

class libqtile.layout.zoomy.**Zoomy**(***config*)

A layout with single active windows, and few other previews at the right

key	default	description
columnwidth	150	Width of the right column
margin	0	Margin of the layout
name	'zoomy'	Name of this layout.
property_big	'1.0'	Property value to set on normal window
property_name	'ZOOM'	Property to set on zoomed window
property_small	'0.1'	Property value to set on zoomed window

4.1.4 Built-in Widgets

AGroupBox

class libqtile.widget.**AGroupBox**(***config*)

A widget that graphically displays the current group

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
border	'000000'	group box border color
borderwidth	3	Current group border width
center_aligned	True	center-aligned group box
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.

Backlight

class libqtile.widget.**Backlight**(***config*)

A simple widget to show the current brightness of a monitor

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
backlight_name	'acpi_video0'	ACPI name of a backlight device
brightness_file	'brightness'	Name of file with the current brightness in /sys/class/backlight/backlight_name
change_command	'xbacklight -set {0}'	Execute command to change value
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'{percent: 2.0%}'	Display format
markup	True	Whether or not to use pango markup
max_brightness_file	'max_brightness'	Name of file with the maximum brightness in /sys/class/backlight/backlight_name
padding	None	Padding. Calculated if None.
step	10	Percent of backlight every scroll changed
update_interval	0.2	The delay in seconds between updates

Battery

class libqtile.widget.**Battery**(***config*)

A text-based battery monitoring widget currently supporting FreeBSD

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
battery	0	Which battery should be monitored (battery number or name)
charge_char	'^'	Character to indicate the battery is charging
discharge_char	'V'	Character to indicate the battery is discharging
empty_char	'x'	Character to indicate the battery is empty
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'{char} {percent:2.0%} {hour:d}:{min:02d} {watt:.2f} W'	Display format
full_char	'='	Character to indicate the battery is full
hide_threshold	None	Hide the text when there is enough energy 0 <= x < 1
low_foreground	'FF0000'	Font color on low battery
low_percentage	0.1	Indicates when to use the low_foreground color 0 < x < 1
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
show_short_text	True	Show "Full" or "Empty" rather than formated text
unknown_char	'?'	Character to indicate the battery status is unknown
update_interval	60	Seconds between status updates

BatteryIcon

class libqtile.widget.**BatteryIcon**(***config*)

Battery life indicator widget.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
battery	0	Which battery should be monitored
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
theme_path	'/home/docs/checkouts/readthedocs.org/user_builds/qtile/checkouts/latest/libqtile/resources/battery-icons'	Path of the icons
update_interval	60	Seconds between status updates

BitcoinTicker

class libqtile.widget.**BitcoinTicker**(***config*)

A bitcoin ticker widget, data provided by the coinbase.com API. Defaults to displaying currency in whatever the current locale is. Examples:

```
# display the average price of bitcoin in local currency
widget.BitcoinTicker()

# display it in Euros:
widget.BitcoinTicker(currency="EUR")
```

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
currency	''	The currency the value that bitcoin is displayed in
data	None	Post Data
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
headers	{}	Extra Headers
json	True	Is Json?
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
parse	None	Parse Function
update_interval	600	Update interval in seconds, if none, the widget updates whenever the event loop is idle.
url	None	Url
user_agent	'Qtile'	Set the user agent
xml	False	Is XML?

CPU

class libqtile.widget.**CPU**(**config*)

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'CPU {freq_current}GHz {load_percent}%'	CPU display format
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
update_interval	1.0	Update interval for the CPU widget

CPUGraph

class libqtile.widget.**CPUGraph**(**config*)

Display CPU usage graph

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
border_color	'215578'	Widget border color
border_width	2	Widget border width
core	'all'	Which core to show (all/0/1/2/...)
fill_color	'1667EB.3'	Fill color for linefill graph
frequency	1	Update frequency in seconds
graph_color	'18BAEB'	Graph color
line_width	3	Line width
margin_x	3	Margin X
margin_y	3	Margin Y
samples	100	Count of graph samples.
start_pos	'bottom'	Drawer starting position ('bottom'/'top')
type	'linefill'	'box', 'line', 'linefill'

Canto

class libqtile.widget.**Canto**(**_config_)

Display RSS feeds updates using the canto console reader

Supported bar orientations: horizontal only

key	default	description
all_format	'{number}'	All feeds display format
background	None	Widget background color
feeds	[]	List of feeds to display, empty for all
fetch	False	Whether to fetch new items on update
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
one_format	'{name}: {number}'	One feed display format
padding	None	Padding. Calculated if None.
update_interval	600	Update interval in seconds, if none, the widget updates whenever the event loop is idle.

CapsNumLockIndicator

class libqtile.widget.**CapsNumLockIndicator**(**_config_)

Really simple widget to show the current Caps/Num Lock state.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
update_interval	0.5	Update Time in seconds.

CheckUpdates

class libqtile.widget.**CheckUpdates**(***config*)

Shows number of pending updates in different unix systems

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
colour_have_updates	'ffffff'	Colour when there are updates.
colour_no_updates	'ffffff'	Colour when there's no updates.
custom_command	None	Custom shell command for checking updates (counts the lines of the output)
display_format	'Updates: {updates}'	Display format if updates available
distro	'Arch'	Name of your distribution
execute	None	Command to execute on click
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
restart_indicator	'ffffff'	Indicator to represent reboot is required. (Ubuntu only)
update_interval	60	Update interval in seconds.

Clipboard

class libqtile.widget.**Clipboard**(*width=CALCULATED*, ***config*)

Display current clipboard contents

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
blacklist	['keepassx']	list with blacklisted wm_class, sadly not every clipboard window sets them, keepassx does.Clipboard contents from blacklisted wm_classes will be replaced by the value of blacklist_text.
blacklist_text	'**********'	text to display when the wm_class is blacklisted
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
max_width	10	maximum number of characters to display (None for all, useful when width is bar.STRETCH)
padding	None	Padding. Calculated if None.
selection	'CLIPBOARD'	the selection to display(CLIPBOARD or PRIMARY)
timeout	10	Default timeout (seconds) for display text, None to keep forever

Clock

class libqtile.widget.**Clock**(**_config_)

A simple but flexible text-based clock

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'%H:%M'	A Python datetime format string
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
timezone	None	The timezone to use for this clock, either as string if pytz is installed (e.g. "US/Central" or anything in /usr/share/zoneinfo), or as tzinfo (e.g. datetime.timezone.utc). None means the system local timezone and is the default.
update_interval	1.0	Update interval for the clock

Cmus

class libqtile.widget.**Cmus**(**_config_)

A simple Cmus widget.

Show the artist and album of now listening song and allow basic mouse control from the bar:

- toggle pause (or play if stopped) on left click;
- skip forward in playlist on scroll up;

- skip backward in playlist on scroll down.

Cmus (https://cmus.github.io) should be installed.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
max_chars	0	Maximum number of characters to display in widget.
noplay_color	'cecece'	Text colour when not playing.
padding	None	Padding. Calculated if None.
play_color	'00ff00'	Text colour when playing.
update_interval	0.5	Update Time in seconds.

Countdown

class libqtile.widget.**Countdown**(***config*)

A simple countdown timer text widget

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
date	datetime.datetime(2020, 2, 6, 21, 31, 10, 357303)	The datetime for the endo of the countdown
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'{D}d {H}h {M}m {S}s'	Format of the displayed text. Available variables:{D} == days, {H} == hours, {M} == minutes, {S} seconds.
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
update_interval	1.0	Update interval in seconds for the clock

CurrentLayout

class libqtile.widget.**CurrentLayout**(*width=CALCULATED*, ***config*)

Display the name of the current layout of the current group of the screen, the bar containing the widget, is on.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.

CurrentLayoutIcon

class libqtile.widget.**CurrentLayoutIcon**(***config*)

Display the icon representing the current layout of the current group of the screen on which the bar containing the widget is.

If you are using custom layouts, a default icon with question mark will be displayed for them. If you want to use custom icon for your own layout, for example, *FooGrid*, then create a file named "layout-foogrid.png" and place it in *~/.icons* directory. You can as well use other directories, but then you need to specify those directories in *custom_icon_paths* argument for this plugin.

The order of icon search is:

- dirs in *custom_icon_paths* config argument
- *~/.icons*
- built-in qtile icons

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
custom_icon_paths	[]	List of folders where to search icons beforeusing built-in icons or icons in ~/.icons dir. This can also be used to providemissing icons for custom layouts. Defaults to empty list.
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
scale	1	Scale factor relative to the bar height. Defaults to 1

CurrentScreen

class libqtile.widget.**CurrentScreen**(*width=CALCULATED*, ***config*)

Indicates whether the screen this widget is on is currently active or not

Supported bar orientations: horizontal only

key	default	description
active_color	'00ff00'	Color when screen is active
active_text	'A'	Text displayed when the screen is active
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
inactive_color	'ff0000'	Color when screen is inactive
inactive_text	'I'	Text displayed when the screen is inactive
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.

DF

class libqtile.widget.**DF**(***config*)

Disk Free Widget

By default the widget only displays if the space is less than warn_space.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'{p} ({uf}{m}|{r:.0f}%)'	String format (p: partition, s: size, f: free space, uf: user free space, m: measure, r: ratio (uf/s))
markup	True	Whether or not to use pango markup
measure	'G'	Measurement (G, M, B)
padding	None	Padding. Calculated if None.
partition	'/'	the partition to check space
update_interval	60	The update interval.
visible_on_warn	True	Only display if warning
warn_color	'ff0000'	Warning color
warn_space	2	Warning space in scale defined by the measure option.

DebugInfo

class libqtile.widget.**DebugInfo**(***config*)

Displays debugging infos about selected window

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.

GenPollText

class libqtile.widget.**GenPollText**(**_config_)

A generic text widget that polls using poll function to get the text

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
func	None	Poll Function
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
update_interval	600	Update interval in seconds, if none, the widget updates whenever the event loop is idle.

GenPollUrl

class libqtile.widget.**GenPollUrl**(**_config_)

A generic text widget that polls an url and parses it using parse function

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
data	None	Post Data
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
headers	{}	Extra Headers
json	True	Is Json?
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
parse	None	Parse Function
update_interval	600	Update interval in seconds, if none, the widget updates whenever the event loop is idle.
url	None	Url
user_agent	'Qtile'	Set the user agent
xml	False	Is XML?

GmailChecker

class libqtile.widget.**GmailChecker**(***config*)

A simple gmail checker. If 'status_only_unseen' is True - set 'fmt' for one argument, ex. 'unseen: {0}'

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
email_path	'INBOX'	email_path
fmt	'inbox[{0}], unseen[{1}]'	fmt
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
password	None	password
status_only_unseen	False	Only show unseen messages
update_interval	30	Update time in seconds.
username	None	username

GroupBox

class libqtile.widget.**GroupBox**(***config*)

A widget that graphically displays the current group. All groups are displayed by their label. If the label of a group is the empty string that group will not be displayed.

Supported bar orientations: horizontal only

key	default	description
active	'FFFFFF'	Active group font colour
background	None	Widget background color
borderwidth	3	Current group border width
center_aligned	True	center-aligned group box
disable_drag	False	Disable dragging and dropping of group names on widget
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
hide_unused	False	Hide groups that have no windows and that are not displayed on any screen.
highlight_color	['000000', '282828']	Active group highlight color when using 'line' highlight method.
highlight_method	'border'	Method of highlighting ('border', 'block', 'text', or 'line')Uses *_*border* color settings
inactive	'404040'	Inactive group font colour
invert_mouse_wheel	False	Whether to invert mouse wheel group movement
markup	True	Whether or not to use pango markup
other_current_screen_border	'404040'	Border or line colour for group on other screen when focused.
other_screen_border	'404040'	Border or line colour for group on other screen when unfocused.
padding	None	Padding. Calculated if None.
rounded	True	To round or not to round box borders
spacing	None	Spacing between groups(if set to None, will be equal to margin_x)
this_current_screen_border	'215578'	Border or line colour for group on this screen when focused.
this_screen_border	'215578'	Border or line colour for group on this screen when unfocused.
urgent_alert_method	'border'	Method for alerting you of WM urgent hints (one of 'border', 'text', 'block', or 'line')
urgent_border	'FF0000'	Urgent border or line color
urgent_text	'FF0000'	Urgent group font color
use_mouse_wheel	True	Whether to use mouse wheel events
visible_groups	None	Groups that will be visible. If set to None or [], all groups will be visible.Visible groups are identified by name not by their displayed label.

HDDBusyGraph

class libqtile.widget.**HDDBusyGraph**(**_config_)

Display HDD busy time graph

Parses /sys/block/<dev>/stat file and extracts overall device IO usage, based on `io_ticks`'s value. See https://www.kernel.org/doc/Documentation/block/stat.txt

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
border_color	'215578'	Widget border color
border_width	2	Widget border width
device	'sda'	Block device to display info for
fill_color	'1667EB.3'	Fill color for linefill graph
frequency	1	Update frequency in seconds
graph_color	'18BAEB'	Graph color
line_width	3	Line width
margin_x	3	Margin X
margin_y	3	Margin Y
samples	100	Count of graph samples.
start_pos	'bottom'	Drawer starting position ('bottom'/'top')
type	'linefill'	'box', 'line', 'linefill'

HDDGraph

class libqtile.widget.**HDDGraph**(***config*)

Display HDD free or used space graph

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
border_color	'215578'	Widget border color
border_width	2	Widget border width
fill_color	'1667EB.3'	Fill color for linefill graph
frequency	1	Update frequency in seconds
graph_color	'18BAEB'	Graph color
line_width	3	Line width
margin_x	3	Margin X
margin_y	3	Margin Y
path	'/'	Partition mount point.
samples	100	Count of graph samples.
space_type	'used'	free/used
start_pos	'bottom'	Drawer starting position ('bottom'/'top')
type	'linefill'	'box', 'line', 'linefill'

IdleRPG

class libqtile.widget.**IdleRPG**(***config*)

A widget for monitoring and displaying IdleRPG stats.

```
# display idlerpg stats for the player 'pants' on freenode's #idlerpg
widget.IdleRPG(url="http://xethron.lolhosting.net/xml.php?player=pants")
```

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
data	None	Post Data
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'IdleRPG: {online} TTL: {ttl}'	Display format
headers	{}	Extra Headers
json	False	Not json :)
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
parse	None	Parse Function
update_interval	600	Update interval in seconds, if none, the widget updates whenever the event loop is idle.
url	None	Url
user_agent	'Qtile'	Set the user agent
xml	True	Is XML :)

Image

class libqtile.widget.**Image**(*length=CALCULATED*, *width=None*, ***config*)

Display a PNG image on the bar

Supported bar orientations: horizontal and vertical

key	default	description
background	None	Widget background color
filename	None	Image filename. Can contain '~'
rotate	0.0	rotate the image in degrees counter-clockwise
scale	True	Enable/Disable image scaling

ImapWidget

class libqtile.widget.**ImapWidget**(***config*)

Email IMAP widget

This widget will scan one of your imap email boxes and report the number of unseen messages present. I've configured it to only work with imap with ssl. Your password is obtained from the Gnome Keyring.

Writing your password to the keyring initially is as simple as (changing out <userid> and <password> for your userid and password):

1) create the file ~/.local/share/python_keyring/keyringrc.cfg with the following contents:

```
[backend]
default-keyring=keyring.backends.Gnome.Keyring
keyring-path=/home/<userid>/.local/share/keyring/
```

2) Execute the following python shell script once:

```
#!/usr/bin/env python3
import keyring
user = <userid>
password = <password>
keyring.set_password('imapwidget', user, password)
```

mbox names must include the path to the mbox (except for the default INBOX). So, for example if your mailroot is `~/Maildir`, and you want to look at the mailbox at HomeMail/fred, the mbox setting would be: `mbox="~/Maildir/HomeMail/fred"`. Note the nested sets of quotes! Labels can be whatever you choose, of course.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
label	'INBOX'	label for display
markup	True	Whether or not to use pango markup
mbox	'"INBOX"'	mailbox to fetch
padding	None	Padding. Calculated if None.
server	None	email server name
update_interval	600	Update interval in seconds, if none, the widget updates whenever the event loop is idle.
user	None	email username

KeyboardKbdd

class libqtile.widget.**KeyboardKbdd**(***config*)

Widget for changing keyboard layouts per window, using kbdd

kbdd should be installed and running, you can get it from: https://github.com/qnikst/kbdd

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
colours	None	foreground colour for each layouteither 'None' or a list of colours.example: ['ffffff', 'E6F0AF'].
configured_keyboards	['us', 'ir']	your predefined list of keyboard layouts.example: ['us', 'ir', 'es']
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
update_interval	1	Update interval in seconds.

KeyboardLayout

class libqtile.widget.**KeyboardLayout**(***config*)
Widget for changing and displaying the current keyboard layout

It requires setxkbmap to be available in the system.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
configured_keyboards	['us']	A list of predefined keyboard layouts represented as strings. For example: ['us', 'us colemak', 'es', 'fr'].
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
option	None	string of setxkbmap option. Ex., 'compose:menu,grp_led:scroll'
padding	None	Padding. Calculated if None.
update_interval	1	Update time in seconds.

KhalCalendar

class libqtile.widget.**KhalCalendar**(***config*)
Khal calendar widget

This widget will display the next appointment on your Khal calendar in the qtile status bar. Appointments within the "reminder" time will be highlighted.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'FFFF33'	default foreground color
lookahead	7	days to look ahead in the calendar
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
reminder_color	'FF0000'	color of calendar entries during reminder time
remindertime	10	reminder time in minutes
update_interval	600	Update interval in seconds, if none, the widget updates whenever the event loop is idle.

LaunchBar

class libqtile.widget.**LaunchBar**(*progs=None*, *width=CALCULATED*, ***config*)
A widget that display icons to launch the associated command

Parameters

progs : a list of tuples `(software_name, command_to_execute, comment)`, for example:

```
('thunderbird', 'thunderbird -safe-mode', 'launch thunderbird in␣
↪safe mode')
('logout', 'qshell:self.qtile.cmd_shutdown()', 'logout from qtile')
```

Supported bar orientations: horizontal only

key	default	description
`background`	`None`	Widget background color
`default_icon`	`'/usr/share/icons/oxygen/256x256/mimetypes/application-x-executable.png'`	Default icon not found
`padding`	`2`	Padding between icons

Maildir

class libqtile.widget.**Maildir**(***config*)

A simple widget showing the number of new mails in maildir mailboxes

Supported bar orientations: horizontal only

key	default	description
`background`	`None`	Widget background color
`fmt`	`'{}'`	How to format the text
`font`	`'sans'`	Default font
`fontshadow`	`None`	font shadow color, default is None(no shadow)
`fontsize`	`None`	Font size. Calculated if None.
`foreground`	`'ffffff'`	Foreground colour
`maildir_path`	`'~/Mail'`	path to the Maildir folder
`markup`	`True`	Whether or not to use pango markup
`padding`	`None`	Padding. Calculated if None.
`separator`	`' '`	the string to put between the subfolder strings.
`sub_folders`	`[]`	The subfolders to scan (e.g. [{"path": "INBOX", "label": "Home mail"}, {"path": "spam", "label": "Home junk"}]
`total`	`False`	Whether or not to sum subfolders into a grand total. The first label will be used.
`update_interval`	`1600`	Update interval in seconds, if none, the widget updates whenever the event loop is idle.

Memory

class libqtile.widget.**Memory**(***config*)

Displays memory/swap usage

MemUsed: Returns memory in use MemTotal: Returns total amount of memory MemFree: Returns amount of memory free Buffers: Returns buffer amount Active: Returns active memory Inactive: Returns inactive memory Shmem: Returns shared memory SwapTotal: Returns total amount of swap SwapFree: Returns amount of swap free SwapUsed: Returns amount of swap in use

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'{MemUsed}M/{MemTotal}M'	Formatting for field names.
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
update_interval	1.0	Update interval for the Memory

MemoryGraph

class libqtile.widget.**MemoryGraph**(**config*)

Displays a memory usage graph

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
border_color	'215578'	Widget border color
border_width	2	Widget border width
fill_color	'1667EB.3'	Fill color for linefill graph
frequency	1	Update frequency in seconds
graph_color	'18BAEB'	Graph color
line_width	3	Line width
margin_x	3	Margin X
margin_y	3	Margin Y
samples	100	Count of graph samples.
start_pos	'bottom'	Drawer starting position ('bottom'/'top')
type	'linefill'	'box', 'line', 'linefill'

Moc

class libqtile.widget.**Moc**(**config*)

A simple MOC widget.

Show the artist and album of now listening song and allow basic mouse control from the bar:

- toggle pause (or play if stopped) on left click;
- skip forward in playlist on scroll up;
- skip backward in playlist on scroll down.

MOC (http://moc.daper.net) should be installed.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
max_chars	0	Maximum number of characters to display in widget.
noplay_color	'cecece'	Text colour when not playing.
padding	None	Padding. Calculated if None.
play_color	'00ff00'	Text colour when playing.
update_interval	0.5	Update Time in seconds.

Mpd

class libqtile.widget.**Mpd**(**config*)

A widget for the Music Player Daemon (MPD)

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
do_color_progress	True	Whether to indicate progress in song by altering message color
fmt	'{}'	How to format the text
fmt_playing	'%a - %t [%v%%]'	Format string to display when playing/paused
fmt_stopped	'Stopped [%v%%]'	Format strings to display when stopped
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
foreground_progress	'ffffff'	Foreground progress colour
host	'localhost'	Host to connect to, can be either an IP address or a UNIX socket path
markup	True	Whether or not to use pango markup
msg_nc	'Mpd off'	Which message to show when we're not connected
padding	None	Padding. Calculated if None.
password	None	Password to use
port	6600	Port to connect to
reconnect	False	Attempt to reconnect if initial connection failed
reconnect_interval	1	Time to delay between connection attempts.
update_interval	0.5	Update Time in seconds.

Mpd2

class libqtile.widget.**Mpd2**(*status_format='{play_status} {artist}/{title} [{repeat}{random}{single}{consume}{updating_db}]'*, *prepare_status={'consume': <function option.<locals>._convert>, 'random': <function option.<locals>._convert>, 'repeat': <function option.<locals>._convert>, 'single': <function option.<locals>._convert>, 'updating_db': <function option.<locals>._convert>}*, ***config*)

A widget for Music Player Daemon (MPD) based on python-mpd2

This widget exists since python-mpd library is no more supported.

Parameters

status_format : format string to display status

Full list of values see in `status` and `currentsong` commands

https://musicpd.org/doc/protocol/command_reference.html#command_status https://musicpd.org/doc/protocol/tags.html

Default:

```
{play_status} {artist}/{title} [{repeat}{random}{single}{consume}
↪{updating_db}]
```

`play_status` is string from `play_states` dict

Note that `time` property of song renamed to `fulltime` to prevent conflicts with status information during formating.

prepare_status : dict of functions for replace values in status with custom

```
f(status, key, space_element) => str
```

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
command	None	Executable command by "command" shortcut
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
host	'localhost'	Host of mpd server
idletimeout	5	MPDClient idle command timeout
keys	{'command': None, 'next': 5, 'previous': 4, 'stop': 3, 'toggle': 1}	Shortcut keys
markup	True	Whether or not to use pango markup
no_connection	'No connection'	Text when mpd is disconnected
padding	None	Padding. Calculated if None.
password	None	Password for auth on mpd server
play_states	{'pause': '', 'play': '', 'stop': ''}	Play state mapping
port	6600	Port of mpd server
space	'-'	Space keeper
timeout	30	MPDClient timeout
update_interval	1	Interval of update widget

Mpris

class libqtile.widget.**Mpris**(***config*)

MPRIS player widget

A widget which displays the current track/artist of your favorite MPRIS player. It should work with all players which implement a reasonably correct version of MPRIS, though I have only tested it with clementine.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
name	'clementine'	Name of the widget
objname	'org.mpris.clementine'	DBUS object to connect to
padding	None	Padding. Calculated if None.
stop_pause_text	'Stopped'	Optional text to display when in the stopped/paused state

Mpris2

class libqtile.widget.**Mpris2**(***config*)

An MPRIS 2 widget

A widget which displays the current track/artist of your favorite MPRIS player. It should work with all MPRIS 2 compatible players which implement a reasonably correct version of MPRIS, though I have only tested it with audacious. This widget scrolls the text if neccessary and information that is displayed is configurable.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
display_metadata	['xesam:title', 'xesam:album', 'xesam:artist']	Which metadata identifiers to display. See http://www.freedesktop.org/wiki/Specifications/mpris-spec/metadata/#index5h3 for available values
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
name	'audacious'	Name of the MPRIS widget.
objname	'org.mpris.MediaPlayer2.audacious'	DBUS MPRIS 2 compatible player identifier- Find it out with dbus-monitor - Also see: http://specifications.freedesktop.org/mpris-spec/latest/#Bus-Name-Policy
padding	None	Padding. Calculated if None.
scroll_chars	30	How many chars at once to display.
scroll_interval	0.5	Scroll delay interval.
scroll_wait_intervals	8	Wait x scroll_interval beforescrolling/removing text
stop_pause_text	None	Optional text to display when in the stopped/paused state

Net

class libqtile.widget.**Net**(**config*)
Displays interface down and up speed

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'{interface}: {down} ↓↑ {up}'	Display format of down-/upload speed of given interfaces
interface	None	List of interfaces or single NIC as string to monitor, None to displays all active NICs combined
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
update_interval	1	The update interval.
use_bits	False	Use bits instead of bytes per second?

NetGraph

class libqtile.widget.**NetGraph**(**config*)
Display a network usage graph

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
bandwidth_type	'down'	down(load)/up(load)
border_color	'215578'	Widget border color
border_width	2	Widget border width
fill_color	'1667EB.3'	Fill color for linefill graph
frequency	1	Update frequency in seconds
graph_color	'18BAEB'	Graph color
interface	'auto'	Interface to display info for ('auto' for detection)
line_width	3	Line width
margin_x	3	Margin X
margin_y	3	Margin Y
samples	100	Count of graph samples.
start_pos	'bottom'	Drawer starting position ('bottom'/'top')
type	'linefill'	'box', 'line', 'linefill'

Notify

class libqtile.widget.**Notify**(*width=CALCULATED*, **config*)
A notify widget

Supported bar orientations: horizontal only

key	default	description
audiofile	None	Audiofile played during notifications
background	None	Widget background color
default_timeout	None	Default timeout (seconds) for notifications
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
foreground_low	'dddddd'	Foreground low priority colour
foreground_urgent	'ff0000'	Foreground urgent priority colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.

Pacman

class libqtile.widget.**Pacman**(***config*)

Shows number of available updates

Needs the pacman package manager installed. So will only work in Arch Linux installation.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
execute	None	Command to execute on click
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
unavailable	'ffffff'	Unavailable Color - no updates.
update_interval	60	The update interval.

Pomodoro

class libqtile.widget.**Pomodoro**(***config*)

Pomodoro technique widget

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
color_active	'00ff00'	Colour then pomodoro is running
color_break	'ffff00'	Colour then it is break time
color_inactive	'ff0000'	Colour then pomodoro is inactive
fmt	'{}'	fmt
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
length_long_break	15	Length of a long break in minutes
length_pomodori	25	Length of one pomodori in minutes
length_short_break	5	Length of a short break in minutes
markup	True	Whether or not to use pango markup
notification_on	True	Turn notifications on
num_pomodori	4	Number of pomodori to do in a cycle
padding	None	Padding. Calculated if None.
prefix_active	''	Prefix then app is active
prefix_break	'B '	Prefix during short break
prefix_inactive	'POMODORO'	Prefix when app is inactive
prefix_long_break	'LB '	Prefix during long break
prefix_paused	'PAUSE'	Prefix during pause
update_interval	1	Update interval in seconds, if none, the widget updates whenever the event loop is idle.

Prompt

class libqtile.widget.**Prompt**(*name='prompt'*, ***config*)

A widget that prompts for user input

Input should be started using the `.start_input()` method on this class.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
bell_style	'audible'	Alert at the begin/end of the command history. Possible values: 'audible', 'visual' and None.
cursor	True	Show a cursor
cursor_color	'bef098'	Color for the cursor and text over it.
cursorblink	0.5	Cursor blink rate. 0 to disable.
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
ignore_dups_history	False	Don't store duplicates in history
markup	True	Whether or not to use pango markup
max_history	100	Commands to keep in history. 0 for no limit.
padding	None	Padding. Calculated if None.
prompt	'{prompt}: '	Text displayed at the prompt
record_history	True	Keep a record of executed commands
visual_bell_color	'ff0000'	Color for the visual bell (changes prompt background).
visual_bell_time	0.2	Visual bell duration (in seconds).

QuickExit

class libqtile.widget.**QuickExit**(*widget=CALCULATED*, ***config*)

A button of exiting the running qtile easily. When clicked this button, a countdown start. If the button pushed with in the countdown again, the qtile shutdown.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
countdown_format	'[{} seconds]'	This text is showed when counting down.
countdown_start	5	Time to accept the second pushing.
default_text	'[shutdown]'	A text displayed as a button
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
timer_interval	1	A countdown interval.

Sep

class libqtile.widget.**Sep**(*height_percent=None*, ***config*)

A visible widget separator

Supported bar orientations: horizontal and vertical

key	default	description
background	None	Widget background color
foreground	'888888'	Separator line colour.
linewidth	1	Width of separator line.
padding	2	Padding on either side of separator.
size_percent	80	Size as a percentage of bar size (0-100).

She

class libqtile.widget.**She**(***config*)

Widget to display the Super Hybrid Engine status

Can display either the mode or CPU speed on eeepc computers.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
device	'/sys/devices/platform/eeepc/cpufv'	sys path to cpufv
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'speed'	Type of info to display "speed" or "name"
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
update_interval	0.5	Update Time in seconds.

Spacer

class libqtile.widget.**Spacer**(*length=STRETCH*, *width=None*, ***config*)

Just an empty space on the bar

Often used with length equal to bar.STRETCH to push bar widgets to the right or bottom edge of the screen.

Parameters

length : Length of the widget. Can be either `bar.STRETCH` or a length in pixels.

width : DEPRECATED, same as `length`.

Supported bar orientations: horizontal and vertical

key	default	description
background	None	Widget background color

StockTicker

class libqtile.widget.**StockTicker**(***config*)

A stock ticker widget, based on the alphavantage API. Users must acquire an API key from https://www.alphavantage.co/support/#api-key

The widget defaults to the TIME_SERIES_INTRADAY API function (i.e. stock symbols), but arbitrary Alpha Vantage API queries can be made by passing extra arguments to the constructor.

```
# Display AMZN
widget.StockTicker(apikey=..., symbol="AMZN")

# Display BTC
widget.StockTicker(apikey=..., function="DIGITAL_CURRENCY_INTRADAY", symbol="BTC",
↪ market="USD")
```

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
data	None	Post Data
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
function	'TIME_SERIES_INTRADAY'	The default API function to query
headers	{}	Extra Headers
interval	'1min'	The default latency to query
json	True	Is Json?
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
parse	None	Parse Function
update_interval	600	Update interval in seconds, if none, the widget updates whenever the event loop is idle.
url	None	Url
user_agent	'Qtile'	Set the user agent
xml	False	Is XML?

SwapGraph

class libqtile.widget.**SwapGraph**(***config*)

Display a swap info graph

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
border_color	'215578'	Widget border color
border_width	2	Widget border width
fill_color	'1667EB.3'	Fill color for linefill graph
frequency	1	Update frequency in seconds
graph_color	'18BAEB'	Graph color
line_width	3	Line width
margin_x	3	Margin X
margin_y	3	Margin Y
samples	100	Count of graph samples.
start_pos	'bottom'	Drawer starting position ('bottom'/'top')
type	'linefill'	'box', 'line', 'linefill'

Systray

class libqtile.widget.**Systray**(***config*)

A widget that manages system tray

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
icon_size	20	Icon width
padding	5	Padding between icons

TaskList

class libqtile.widget.**TaskList**(***config*)

Displays the icon and name of each window in the current group

Contrary to WindowTabs this is an interactive widget. The window that currently has focus is highlighted.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
border	'215578'	Border colour
borderwidth	2	Current group border width
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
highlight_method	'border'	Method of highlighting (one of 'border' or 'block') Uses *_border color settings
icon_size	None	Icon size. (Calculated if set to None. Icons are hidden if set to 0.)
markup_floating	None	Text markup of the floating window state. Supports pangomarkup with markup=True.e.g., "{}" or "{}"
markup_focused	None	Text markup of the focused window state. Supports pangomarkup with markup=True.e.g., "{}" or "{}"
markup_maximized	None	Text markup of the maximized window state. Supports pangomarkup with markup=True.e.g., "{}" or "{}"
markup_minimized	None	Text markup of the minimized window state. Supports pangomarkup with markup=True.e.g., "{}" or "{}"
markup_normal	None	Text markup of the normal window state. Supports pangomarkup with markup=True.e.g., "{}" or "{}"
max_title_width	None	Max size in pixels of task title.(if set to None, as much as available.)
rounded	True	To round or not to round borders
spacing	None	Spacing between tasks.(if set to None, will be equal to margin_x)
title_width_method	None	Method to compute the width of task title. (None, 'uniform'.)Defaults to None, the normal behaviour.
txt_floating	'V '	Text representation of the floating window state. e.g., "V " or " "
txt_maximized	'[] '	Text representation of the maximized window state. e.g., "[] " or " "
txt_minimized	'_ '	Text representation of the minimized window state. e.g., "_ " or " "
unfocused_border	None	Border color for unfocused windows. Affects only hightlight_method 'border' and 'block'. Defaults to None, which means no special color.
urgent_alert_method	'border'	Method for alerting you of WM urgent hints (one of 'border' or 'text')
urgent_border	'FF0000'	Urgent border color

TextBox

class libqtile.widget.**TextBox**(*text=' '*, *width=CALCULATED*, ***config*)

A flexible textbox that can be updated from bound keys, scripts, and qshell

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Text font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font pixel size. Calculated if None.
foreground	'#ffffff'	Foreground colour.
markup	True	Whether or not to use pango markup
padding	None	Padding left and right. Calculated if None.

ThermalSensor

class libqtile.widget.**ThermalSensor**(***config*)

Widget to display temperature sensor information

For using the thermal sensor widget you need to have lm-sensors installed. You can get a list of the tag_sensors executing "sensors" in your terminal. Then you can choose which you want, otherwise it will display the first available.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
foreground_alert	'ff0000'	Foreground colour alert
markup	True	Whether or not to use pango markup
metric	True	True to use metric/C, False to use imperial/F
padding	None	Padding. Calculated if None.
show_tag	False	Show tag sensor
tag_sensor	None	Tag of the temperature sensor. For example: "temp1" or "Core 0"
threshold	70	If the current temperature value is above, then change to foreground_alert colour
update_interval	2	Update interval in seconds

Volume

class libqtile.widget.**Volume**(***config*)

Widget that display and change volume

If theme_path is set it draw widget as icons.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
cardid	None	Card Id
channel	'Master'	Channel
device	'default'	Device Name
emoji	False	Use emoji to display volume states, only if theme_path is not set.The specified font needs to contain the correct unicode characters.
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
get_volume_command	None	Command to get the current volume
markup	True	Whether or not to use pango markup
mute_command	None	Mute command
padding	3	Padding left and right. Calculated if None.
step	2	Volume change for up an down commands in percentage.Only used if volume_up_command and volume_down_command are not set.
theme_path	None	Path of the icons
update_interval	0.2	Update time in seconds.
volume_app	None	App to control volume
volume_down_command	None	Volume down command
volume_up_command	None	Volume up command

Wallpaper

class libqtile.widget.**Wallpaper**(***config*)

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
directory	'~/Pictures/wallpapers/'	Wallpaper Directory
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
label	None	Use a fixed label instead of image name.
markup	True	Whether or not to use pango markup
one_screen	False	Treat the whole X display as one screen when setting wallpapers (does not work if wallpaper_command is set).
padding	None	Padding. Calculated if None.
random_selection	False	If set, use random initial wallpaper and randomly cycle through the wallpapers.
wallpaper	None	Wallpaper
wallpaper_command	None	Wallpaper command

WindowName

class libqtile.widget.**WindowName**(*width=STRETCH*, ***config*)
Displays the name of the window that currently has focus

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
for_current_screen	False	instead of this bars screen use currently active screen
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
show_state	True	show window status before window name

WindowTabs

class libqtile.widget.**WindowTabs**(***config*)
Displays the name of each window in the current group. Contrary to TaskList this is not an interactive widget. The window that currently has focus is highlighted.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
selected	('<', '>')	Selected task indicator
separator	' | '	Task separator text.

Wlan

class libqtile.widget.**Wlan**(***config*)
Displays Wifi SSID and quality.

Widget requirements: iwlib.

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
disconnected_message	'Disconnected'	String to show when the wlan is diconnected.
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'{essid} {quality}/70'	Display format. For percents you can use "{essid} {percent:2.0%}"
interface	'wlan0'	The interface to monitor
markup	True	Whether or not to use pango markup
padding	None	Padding. Calculated if None.
update_interval	1	The update interval.

YahooWeather

class libqtile.widget.**YahooWeather**(***config*)

A weather widget, data provided by the Yahoo! Weather API.

Format options:

- astronomy_sunrise
- astronomy_sunset
- atmosphere_humidity
- atmosphere_visibility
- atmosphere_pressure
- atmosphere_rising
- condition_text
- condition_code
- condition_temp
- condition_date
- location_city
- location_region
- location_country
- units_temperature
- units_distance
- units_pressure
- units_speed
- wind_chill

Supported bar orientations: horizontal only

key	default	description
background	None	Widget background color
data	None	Post Data
down	'v'	symbol for falling atmospheric pressure
fmt	'{}'	How to format the text
font	'sans'	Default font
fontshadow	None	font shadow color, default is None(no shadow)
fontsize	None	Font size. Calculated if None.
foreground	'ffffff'	Foreground colour
format	'{location_city}: {condition_temp} °{units_temperature}'	Display format
headers	{}	Extra Headers
json	True	Is Json?
location	None	Location to fetch weather for. Ignored if woeid is set.
markup	True	Whether or not to use pango markup
metric	True	True to use metric/C, False to use imperial/F
padding	None	Padding. Calculated if None.
parse	None	Parse Function
steady	's'	symbol for steady atmospheric pressure
up	'^'	symbol for rising atmospheric pressure
update_interval	600	Update interval in seconds, if none, the widget updates whenever the event loop is idle.
url	None	Url
user_agent	'Qtile'	Set the user agent
woeid	None	Where On Earth ID. Auto-calculated if location is set.
xml	False	Is XML?

4.1.5 Built-in Extensions

CommandSet

class libqtile.extension.**CommandSet**(***config*)

Give list of commands to be executed in dmenu style.

ex. manage mocp deamon:

```
Key([mod], 'm', lazy.run_extension(extension.CommandSet(
    commands={
        'play/pause': '[ $(mocp -i | wc -l) -lt 2 ] && mocp -p || mocp -G',
        'next': 'mocp -f',
        'previous': 'mocp -r',
        'quit': 'mocp -x',
        'open': 'urxvt -e mocp',
        'shuffle': 'mocp -t shuffle',
        'repeat': 'mocp -t repeat',
        },
    pre_commands=['[ $(mocp -i | wc -l) -lt 1 ] && mocp -S'],
    **Theme.dmenu))),
```

key	default	description
background	None	defines the normal background color
command	None	the command to be launched (string or list with arguments)
commands	None	dictionary of commands where key is runable command
dmenu_bottom	False	dmenu appears at the bottom of the screen
dmenu_command	'dmenu'	the dmenu command to be launched
dmenu_font	None	override the default 'font' and 'fontsize' options for dmenu
dmenu_height	None	defines the height (only supported by some dmenu forks)
dmenu_ignorecase	False	dmenu matches menu items case insensitively
dmenu_lines	None	dmenu lists items vertically, with the given number of lines
dmenu_prompt	None	defines the prompt to be displayed to the left of the input field
font	'sans'	defines the font name to be used
fontsize	None	defines the font size to be used
foreground	None	defines the normal foreground color
pre_commands	None	list of commands to be executed before getting dmenu answer
selected_background	None	defines the selected background color
selected_foreground	None	defines the selected foreground color

Dmenu

class libqtile.extension.**Dmenu**(***config*)

Python wrapper for dmenu http://tools.suckless.org/dmenu/

key	default	description
background	None	defines the normal background color
command	None	the command to be launched (string or list with arguments)
dmenu_bottom	False	dmenu appears at the bottom of the screen
dmenu_command	'dmenu'	the dmenu command to be launched
dmenu_font	None	override the default 'font' and 'fontsize' options for dmenu
dmenu_height	None	defines the height (only supported by some dmenu forks)
dmenu_ignorecase	False	dmenu matches menu items case insensitively
dmenu_lines	None	dmenu lists items vertically, with the given number of lines
dmenu_prompt	None	defines the prompt to be displayed to the left of the input field
font	'sans'	defines the font name to be used
fontsize	None	defines the font size to be used
foreground	None	defines the normal foreground color
selected_background	None	defines the selected background color
selected_foreground	None	defines the selected foreground color

DmenuRun

class libqtile.extension.**DmenuRun**(***config*)

Special case to run applications.

config.py should have something like:

```
from libqtile import extension
keys = [
    Key(['mod4'], 'r', lazy.run_extension(extension.DmenuRun(
        dmenu_prompt=">",
```

(continues on next page)

(continued from previous page)

```
        dmenu_font="Andika-8",
        background="#15181a",
        foreground="#00ff00",
        selected_background="#079822",
        selected_foreground="#fff",
        dmenu_height=24,  # Only supported by some dmenu forks
    ))),
]
```

key	default	description
background	None	defines the normal background color
command	None	the command to be launched (string or list with arguments)
dmenu_bottom	False	dmenu appears at the bottom of the screen
dmenu_command	'dmenu_run'	the dmenu command to be launched
dmenu_font	None	override the default 'font' and 'fontsize' options for dmenu
dmenu_height	None	defines the height (only supported by some dmenu forks)
dmenu_ignorecase	False	dmenu matches menu items case insensitively
dmenu_lines	None	dmenu lists items vertically, with the given number of lines
dmenu_prompt	None	defines the prompt to be displayed to the left of the input field
font	'sans'	defines the font name to be used
fontsize	None	defines the font size to be used
foreground	None	defines the normal foreground color
selected_background	None	defines the selected background color
selected_foreground	None	defines the selected foreground color

J4DmenuDesktop

class libqtile.extension.**J4DmenuDesktop**(***config*)

Python wrapper for j4-dmenu-desktop https://github.com/enkore/j4-dmenu-desktop

key	default	description
background	None	defines the normal background color
command	None	the command to be launched (string or list with arguments)
dmenu_bottom	False	dmenu appears at the bottom of the screen
dmenu_command	'dmenu'	the dmenu command to be launched
dmenu_font	None	override the default 'font' and 'fontsize' options for dmenu
dmenu_height	None	defines the height (only supported by some dmenu forks)
dmenu_ignorecase	False	dmenu matches menu items case insensitively
dmenu_lines	None	dmenu lists items vertically, with the given number of lines
dmenu_prompt	None	defines the prompt to be displayed to the left of the input field
font	'sans'	defines the font name to be used
fontsize	None	defines the font size to be used
foreground	None	defines the normal foreground color
j4dmenu_command	'j4-dmenu-desktop'	the dmenu command to be launched
j4dmenu_display_binary	False	display binary name after each entry
j4dmenu_generic	True	include the generic name of desktop entries
j4dmenu_terminal	None	terminal emulator used to start terminal apps
j4dmenu_usage_log	None	file used to sort items by usage frequency
j4dmenu_use_xdg_de	False	read $XDG_CURRENT_DESKTOP to determine the desktop environment
selected_background	None	defines the selected background color
selected_foreground	None	defines the selected foreground color

RunCommand

class libqtile.extension.**RunCommand**(***config*)

Run an arbitrary command.

Mostly useful as a superclass for more specific extensions that need to interact with the qtile object.

Also consider simply using lazy.spawn() or writing a client.

key	default	description
background	None	defines the normal background color
command	None	the command to be launched (string or list with arguments)
font	'sans'	defines the font name to be used
fontsize	None	defines the font size to be used
foreground	None	defines the normal foreground color
selected_background	None	defines the selected background color
selected_foreground	None	defines the selected foreground color

WindowList

class libqtile.extension.**WindowList**(***config*)

Give vertical list of all open windows in dmenu. Switch to selected.

key	default	description
all_groups	True	If True, list windows from all groups; otherwise only from the current group
background	None	defines the normal background color
command	None	the command to be launched (string or list with arguments)
dmenu_bottom	False	dmenu appears at the bottom of the screen
dmenu_command	'dmenu'	the dmenu command to be launched
dmenu_font	None	override the default 'font' and 'fontsize' options for dmenu
dmenu_height	None	defines the height (only supported by some dmenu forks)
dmenu_ignorecase	False	dmenu matches menu items case insensitively
dmenu_lines	'80'	Give lines vertically. Set to None get inline
dmenu_prompt	None	defines the prompt to be displayed to the left of the input field
font	'sans'	defines the font name to be used
fontsize	None	defines the font size to be used
foreground	None	defines the normal foreground color
item_format	'{group}.{id}: {window}'	the format for the menu items
selected_background	None	defines the selected background color
selected_foreground	None	defines the selected foreground color

4.2 Frequently Asked Questions

4.2.1 Why the name Qtile?

Users often wonder, why the Q? Does it have something to do with Qt? No. Below is an IRC excerpt where cortesi explains the great trial that ultimately brought Qtile into existence, thanks to the benevolence of the Open Source Gods. Praise be to the OSG!

```
ramnes:  what does Qtile mean?
ramnes:  what's the Q?
@tych0:  ramnes: it doesn't :)
@tych0:  cortesi was just looking for the first letter that wasn't registered
         in a domain name with "tile" as a suffix
@tych0:  qtile it was :)
cortesi: tych0, dx: we really should have something more compelling to
         explain the name. one day i was swimming at manly beach in sydney,
         where i lived at the time. suddenly, i saw an enormous great white
         right beside me. it went for my leg with massive, gaping jaws, but
         quick as a flash, i thumb-punched it in both eyes. when it reared
         back in agony, i saw that it had a jagged, gnarly scar on its
         stomach... a scar shaped like the letter "Q".
cortesi: while it was distracted, i surfed a wave to shore. i knew that i
         had to dedicate my next open source project to the ocean gods, in
         thanks for my lucky escape. and thus, qtile got its name...
```

4.2.2 When I first start xterm/urxvt/rxvt containing an instance of Vim, I see text and layout corruption. What gives?

Vim is not handling terminal resizes correctly. You can fix the problem by starting your xterm with the "-wf" option, like so:

```
xterm -wf -e vim
```

Alternatively, you can just cycle through your layouts a few times, which usually seems to fix it.

4.2.3 How do I know which modifier specification maps to which key?

To see a list of modifier names and their matching keys, use the `xmodmap` command. On my system, the output looks like this:

```
$ xmodmap
xmodmap:  up to 3 keys per modifier, (keycodes in parentheses):

shift       Shift_L (0x32),  Shift_R (0x3e)
lock        Caps_Lock (0x9)
control     Control_L (0x25),  Control_R (0x69)
mod1        Alt_L (0x40),  Alt_R (0x6c),  Meta_L (0xcd)
mod2        Num_Lock (0x4d)
mod3
mod4        Super_L (0xce),  Hyper_L (0xcf)
mod5        ISO_Level3_Shift (0x5c),  Mode_switch (0xcb)
```

4.2.4 My “pointer mouse cursor” isn’t the one I expect it to be!

Qtile should set the default cursor to left_ptr, you must install xcb-util-cursor if you want support for themed cursors.

4.2.5 LibreOffice menus don’t appear or don’t stay visible

A workaround for problem with the mouse in libreoffice is setting the environment variable »SAL_USE_VCLPLUGIN=gen«. It is dependet on your system configuration where to do this. e.g. Arch-Linux with libreoffice-fresh in /etc/profile.d/libreoffice-fresh.sh.

4.3 License

This project is distributed under the MIT license.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

- genindex

Index

D

E

F

G

H

I

J

K

L

M