rc.lua

```
-- Standard awesome library
require("awful")
require("awful.autofocus")
require("awful.rules")
-- Theme handling library
require("beautiful")
-- Notification library
require("naughty")

-- Load Debian menu entries
require("debian.menu")

-- {{{ Error handling
-- Check if awesome encountered an error during startup and fell back
to
-- another config (This code will only ever execute for the fallback
config)
if awesome.startup_errors then
    naughty.notify({ preset = naughty.config.presets.critical,
                     title = "Oops, there were errors during startup!",
                     text = awesome.startup_errors })
end

-- Handle runtime errors after startup
do
    local in_error = false
    awesome.add_signal("debug::error", function (err)
        -- Make sure we don't go into an endless error loop
        if in_error then return end
        in_error = true

        naughty.notify({ preset = naughty.config.presets.critical,
                         title = "Oops, an error happened!",
                         text = err })
        in_error = false
    end)
end
-- }}}

-- {{{ Variable definitions
-- Themes define colours, icons, and wallpapers
beautiful.init("/usr/share/awesome/themes/zenburn/theme.lua")

-- This is used later as the default terminal and editor to run.
terminal = "x-terminal-emulator"
editor = os.getenv("EDITOR") or "editor"
editor_cmd = terminal .. " -e " .. editor

-- Default modkey.
```

```
-- Usually, Mod4 is the key with a logo between Control and Alt.
-- If you do not like this or do not have such a key,
-- I suggest you to remap Mod4 to another key using xmodmap or other
tools.
-- However, you can use another modifier like Mod1, but it may interact
with others.
modkey = "Mod4"

-- Table of layouts to cover with awful.layout.inc, order matters.
layouts =
{
    awful.layout.suit.tile,
    awful.layout.suit.tile.left,
    awful.layout.suit.tile.bottom,
    awful.layout.suit.tile.top,
    awful.layout.suit.fair,
    awful.layout.suit.fair.horizontal,
--    awful.layout.suit.spiral,
--    awful.layout.suit.spiral.dwindle,
    awful.layout.suit.max,
--    awful.layout.suit.max.fullscreen,
    awful.layout.suit.magnifier,
    awful.layout.suit.floating
}
-- }}}

-- {{{ Tags
-- Define a tag table which hold all screen tags.
tags = {
   names  = { "e", "w", "a", "f", 5, 6, 7, 8, 9 },
   layout = { layouts[1], layouts[9], layouts[1], layouts[5],
layouts[1],
              layouts[1], layouts[1], layouts[1], layouts[1]
 }}
 for s = 1, screen.count() do
     -- Each screen has its own tag table.
     tags[s] = awful.tag(tags.names, s, tags.layout)
 end
 -- }}}

-- {{{ Menu
-- Create a laucher widget and a main menu
myawesomemenu = {
   { "edit config", 'gvim /home/yahya/.config/awesome/rc.lua' },
   { "manual", terminal .. " -e man awesome" },
   --{ "edit config", editor_cmd .. " " .. awesome.conffile },
   { "restart", awesome.restart },
   { "quit", awesome.quit }
}
```

```
mymainmenu = awful.menu({ items = { { "awesome", myawesomemenu,
beautiful.awesome_icon },
                                    --{ "Debian",
debian.menu.Debian_menu.Debian },
                                    { "Chromium", 'chromium' },
                                    { "Iceweasel", 'iceweasel' },
                                    { "Pcmanfm", 'pcmanfm' },
                                    { "Gvim", 'gvim' },
                                    { "Alsamixer", terminal .. ' -e
alsamixer' },
                                    { "Lxappearance", 'lxappearance' },
                                    { "Synaptic", 'gksu synaptic'},
                                    { "Xkill", 'xkill' },
                                    { "Déconnection",
'/home/yahya/.config/awesome/shutdown_dialog.sh'}
                                  }
                        })

mylauncher = awful.widget.launcher({ image =
image(beautiful.awesome_icon),
                                     menu = mymainmenu })
-- }}}

-- {{{ Wibox
-- Create a textclock widget
mytextclock = awful.widget.textclock({ align = "center" })

-- Create a systray
mysystray = widget({ type = "systray" })

-- Create a wibox for each screen and add it
mywibox = {}
mypromptbox = {}
mylayoutbox = {}
mytaglist = {}
mytaglist.buttons = awful.util.table.join(
                    awful.button({ }, 1, awful.tag.viewonly),
                    awful.button({ modkey }, 1,
awful.client.movetotag),
                    awful.button({ }, 3, awful.tag.viewtoggle),
                    awful.button({ modkey }, 3,
awful.client.toggletag),
                    awful.button({ }, 4, awful.tag.viewnext),
                    awful.button({ }, 5, awful.tag.viewprev)
                    )
mytasklist = {}
mytasklist.buttons = awful.util.table.join(
                     awful.button({ }, 1, function (c)
                                              if c == client.focus then
                                                  c.minimized = true
                                              else
```

```
                                                  if not c:isvisible()
then
awful.tag.viewonly(c:tags()[1])
                                                  end
                                                  -- This will also un-
minimize
                                                  -- the client, if
needed
                                                  client.focus = c
                                                  c:raise()
                                              end
                                          end),
                      awful.button({ }, 3, function ()
                                              if instance then
                                                  instance:hide()
                                                  instance = nil
                                              else
                                                  instance =
awful.menu.clients({ width=250 })
                                              end
                                          end),
                      awful.button({ }, 4, function ()
awful.client.focus.byidx(1)
                                              if client.focus then
client.focus:raise() end
                                          end),
                      awful.button({ }, 5, function ()
awful.client.focus.byidx(-1)
                                              if client.focus then
client.focus:raise() end
                                          end))

for s = 1, screen.count() do
    -- Create a promptbox for each screen
    mypromptbox[s] = awful.widget.prompt({ layout =
awful.widget.layout.horizontal.leftright })
    -- Create an imagebox widget which will contains an icon indicating
which layout we're using.
    -- We need one layoutbox per screen.
    mylayoutbox[s] = awful.widget.layoutbox(s)
    mylayoutbox[s]:buttons(awful.util.table.join(
                           awful.button({ }, 1, function ()
awful.layout.inc(layouts, 1) end),
                           awful.button({ }, 3, function ()
awful.layout.inc(layouts, -1) end),
                           awful.button({ }, 4, function ()
awful.layout.inc(layouts, 1) end),
                           awful.button({ }, 5, function ()
awful.layout.inc(layouts, -1) end)))
    -- Create a taglist widget
```

```
    mytaglist[s] = awful.widget.taglist(s,
awful.widget.taglist.label.all, mytaglist.buttons)

    -- Create a tasklist widget
    mytasklist[s] = awful.widget.tasklist(function(c)
                                              return
awful.widget.tasklist.label.currenttags(c, s)
                                          end, mytasklist.buttons)

    -- Create the wibox
    mywibox[s] = awful.wibox({ position = "right", width = "20", screen
= s })
    -- Add widgets to the wibox - order matters
    mywibox[s].widgets = {
        {
            mylauncher,
            mytaglist[s],
            mypromptbox[s],
            layout = awful.widget.layout.horizontal.leftright
        },
        mylayoutbox[s],
        mytextclock,
        s == 1 and mysystray or nil,
        mytasklist[s],
        layout = awful.widget.layout.horizontal.rightleft
    }
end
-- }}}

-- {{{ Mouse bindings
root.buttons(awful.util.table.join(
    awful.button({ }, 3, function () mymainmenu:toggle() end),
    awful.button({ }, 4, awful.tag.viewnext),
    awful.button({ }, 5, awful.tag.viewprev)
))
-- }}}

-- {{{ Key bindings
globalkeys = awful.util.table.join(
    awful.key({ modkey,           }, "Up",   awful.tag.viewprev
),
    awful.key({ modkey,           }, "Down",  awful.tag.viewnext
),
    awful.key({ modkey,           }, "Escape",
awful.tag.history.restore),

    awful.key({ modkey,           }, "j",
        function ()
            awful.client.focus.byidx( 1)
            if client.focus then client.focus:raise() end
        end),
```

```
    awful.key({ modkey,           }, "k",
        function ()
            awful.client.focus.byidx(-1)
            if client.focus then client.focus:raise() end
        end),
    awful.key({ modkey,           }, "/", function ()
mymainmenu:show({keygrabber=true}) end),

    -- Layout manipulation
    awful.key({ modkey, "Shift"   }, "j", function ()
awful.client.swap.byidx(  1)    end),
    awful.key({ modkey, "Shift"   }, "k", function ()
awful.client.swap.byidx( -1)    end),
    awful.key({ modkey, "Control" }, "j", function ()
awful.screen.focus_relative( 1) end),
    awful.key({ modkey, "Control" }, "k", function ()
awful.screen.focus_relative(-1) end),
    awful.key({ modkey,           }, "u", awful.client.urgent.jumpto),
    awful.key({ modkey,           }, "Tab",
        function ()
            awful.client.focus.history.previous()
            if client.focus then
                client.focus:raise()
            end
        end),

    -- Standard program
    awful.key({ modkey,           }, "Return", function ()
awful.util.spawn(terminal) end),
    awful.key({ modkey, "Control" }, "r", awesome.restart),
    awful.key({ modkey, "Shift"   }, "q", awesome.quit),

    awful.key({ modkey,           }, "l",     function ()
awful.tag.incmwfact( 0.05)    end),
    awful.key({ modkey,           }, "h",     function ()
awful.tag.incmwfact(-0.05)    end),
    awful.key({ modkey, "Shift"   }, "h",     function ()
awful.tag.incnmaster( 1)      end),
    awful.key({ modkey, "Shift"   }, "l",     function ()
awful.tag.incnmaster(-1)      end),
    awful.key({ modkey, "Control" }, "h",     function ()
awful.tag.incncol( 1)         end),
    awful.key({ modkey, "Control" }, "l",     function ()
awful.tag.incncol(-1)         end),
    awful.key({ modkey,           }, "space", function ()
awful.layout.inc(layouts,  1) end),
    awful.key({ modkey, "Shift"   }, "space", function ()
awful.layout.inc(layouts, -1) end),

    awful.key({ modkey, "Control" }, "n", awful.client.restore),
```

```
    -- Custom shortcuts
    awful.key({ modkey, "Control"   }, "c", function ()
awful.util.spawn("chromium")      end),
    awful.key({ modkey, "Control"   }, "i", function ()
awful.util.spawn("iceweasel")     end),
    awful.key({ modkey, "Control"   }, "g", function ()
awful.util.spawn("gvim")          end),
    awful.key({ modkey, "Control"   }, "p", function ()
awful.util.spawn("pcmanfm")       end),
    awful.key({ modkey, "Control"   }, "x", function ()
awful.util.spawn("xkill")         end),

    -- Prompt
    awful.key({ modkey },            ";",     function ()
mypromptbox[mouse.screen]:run() end),

    awful.key({ modkey }, "x",
              function ()
                  awful.prompt.run({ prompt = "Run Lua code: " },
                  mypromptbox[mouse.screen].widget,
                  awful.util.eval, nil,
                  awful.util.getdir("cache") .. "/history_eval")
              end)
)

clientkeys = awful.util.table.join(
    awful.key({ modkey,           }, "f",      function (c)
c.fullscreen = not c.fullscreen  end),
    awful.key({ modkey, "Shift"   }, "c",      function (c) c:kill()
end),
    awful.key({ modkey, "Control" }, "space",
awful.client.floating.toggle                     ),
    awful.key({ modkey, "Control" }, "Return", function (c)
c:swap(awful.client.getmaster()) end),
    awful.key({ modkey,           }, "o",
awful.client.movetoscreen                        ),
    awful.key({ modkey, "Shift"   }, "r",      function (c) c:redraw()
end),
    awful.key({ modkey,           }, "t",      function (c) c.ontop =
not c.ontop            end),
    awful.key({ modkey,           }, "n",
        function (c)
            -- The client currently has the input focus, so it cannot
be
            -- minimized, since minimized clients can't have the focus.
            c.minimized = true
        end),
    awful.key({ modkey,           }, "m",
        function (c)
            c.maximized_horizontal = not c.maximized_horizontal
```

```
            c.maximized_vertical   = not c.maximized_vertical
        end)
)

-- Compute the maximum number of digit we need, limited to 9
keynumber = 0
for s = 1, screen.count() do
   keynumber = math.min(9, math.max(#tags[s], keynumber));
end

-- Bind all key numbers to tags.
-- Be careful: we use keycodes to make it works on any keyboard layout.
-- This should map on the top row of your keyboard, usually 1 to 9.
for i = 1, keynumber do
    globalkeys = awful.util.table.join(globalkeys,
        awful.key({ modkey }, "#" .. i + 9,
                  function ()
                        local screen = mouse.screen
                        if tags[screen][i] then
                            awful.tag.viewonly(tags[screen][i])
                        end
                  end),
        awful.key({ modkey, "Control" }, "#" .. i + 9,
                  function ()
                      local screen = mouse.screen
                      if tags[screen][i] then
                          awful.tag.viewtoggle(tags[screen][i])
                      end
                  end),
        awful.key({ modkey, "Shift" }, "#" .. i + 9,
                  function ()
                      if client.focus and tags[client.focus.screen][i]
then
awful.client.movetotag(tags[client.focus.screen][i])
                      end
                  end),
        awful.key({ modkey, "Control", "Shift" }, "#" .. i + 9,
                  function ()
                      if client.focus and tags[client.focus.screen][i]
then
awful.client.toggletag(tags[client.focus.screen][i])
                      end
                  end))
end

clientbuttons = awful.util.table.join(
    awful.button({ }, 1, function (c) client.focus = c; c:raise() end),
    awful.button({ modkey }, 1, awful.mouse.client.move),
    awful.button({ modkey }, 3, awful.mouse.client.resize))
```

```
-- Set keys
root.keys(globalkeys)
-- }}}

-- {{{ Rules
awful.rules.rules = {
    -- All clients will match this rule.
    { rule = { },
      properties = { border_width = beautiful.border_width,
                     border_color = beautiful.border_normal,
                     focus = true,
                     keys = clientkeys,
                     buttons = clientbuttons } },
    { rule = { class = "MPlayer" },
      properties = { floating = true } },
    { rule = { class = "iceweasel" },
      properties = { floating = false, tag = tags[1][2] } },
    { rule = { class = "chromium-browser" },
      properties = { floating = false, tag = tags[1][2] } },
    { rule = { class = "gimp" },
      properties = { floating = true } },
    -- Set Firefox to always map on tags number 2 of screen 1.
    -- { rule = { class = "Firefox" },
    --   properties = { tag = tags[1][2] } },
}
-- }}}

-- {{{ Signals
-- Signal function to execute when a new client appears.
client.add_signal("manage", function (c, startup)
    -- Add a titlebar
    -- awful.titlebar.add(c, { modkey = modkey })

    -- Enable sloppy focus
    c:add_signal("mouse::enter", function(c)
        if awful.layout.get(c.screen) ~= awful.layout.suit.magnifier
            and awful.client.focus.filter(c) then
            client.focus = c
        end
    end)

    if not startup then
        -- Set the windows at the slave,
        -- i.e. put it at the end of others instead of setting it
master.
        -- awful.client.setslave(c)

        -- Put windows in a smart way, only if they does not set an
initial position.
        if not c.size_hints.user_position and not
c.size_hints.program_position then
```

```
            awful.placement.no_overlap(c)
            awful.placement.no_offscreen(c)
        end
    end
end)

client.add_signal("focus", function(c) c.border_color =
beautiful.border_focus end)
client.add_signal("unfocus", function(c) c.border_color =
beautiful.border_normal end)
-- }}}
```