Conky 1.10 - Conky lua Horloge, CPU x6 et Disques

- Objet : conky par l'exemple
- Niveau requis :
  débutant, avisé
- Commentaires : *page à piller.*
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.

- prenez soin d'éditer les fichiers pour les adapter à votre système

Conky lua Horloge, CPU x6 et Disques

Fichier .conkyrc :

.conkyrc

```
conky.config = {
background = false,
update_interval = 1,
cpu_avg_samples = 2,
net_avg_samples = 2,
override_utf8_locale = true,
double_buffer = true,
no_buffers = true,
text_buffer_size = 2048,
temperature_unit = 'fahrenheit',
own_window = true,
own_window_type = 'normal',
own_window_transparent = true,
own_window_argb_visual = true,
own_window_hints = 'undecorated,sticky,skip_taskbar,skip_pager,below',
border_inner_margin = 0,
border_outer_margin = 0,
minimum_width = 200,
minimum_height = 800,
alignment = 'top_right',
gap_x = 35,
gap_y = 55,
draw_shades = false,
draw_outline = false,
draw_borders = false,
```

```
draw_graph_borders = false,
use_xft = true,
font = 'caviar dreams:size=8',
xftalpha = 0.5 ,
uppercase = false,
temperature_unit = 'celsius',
default_color = 'FFFFFF',

lua_load = '~/.conky/clock_rings.lua',
lua_draw_hook_pre = 'clock_rings',

}

conky.text = [[
${voffset 8}${color 1B708D}${font caviar dreams:size=16}${time
%A}${font}${voffset -8}${alignr 0}${color FFFFFF}${font caviar
dreams:bold:size=38}${time %e}${font}
${color FFFFFF}${voffset -30}${color FFFFFF}${font caviar
dreams:size=18}${time %b}${font}${voffset -3} ${color FFFFFF}${font
caviar dreams:size=20}${time %Y}${font}${color 1B708D}${hr}
${image ~/.conky/debian-logo.png -p 76,128 -s 40x40}

${goto 5}${voffset 125}
${font caviar dreams:bold:size=7}${voffset 5}${goto 20}${color
FFFFFF}${freq_g cpu0} Ghz${goto 78}${alignr 330}${cpu cpu0}%
${font caviar dreams:bold:size=6}${voffset 3}${goto 20}${color
1B708D}CPU 1${goto 78}${alignr 330}${color FFFFFF}${cpu cpu1}%
${font caviar dreams:bold:size=6}${voffset 3}${goto 20}${color
1B708D}CPU 2${goto 78}${alignr 330}${color FFFFFF}${cpu cpu2}%
${font caviar dreams:bold:size=6}${voffset 3}${goto 20}${color
1B708D}CPU 3${goto 78}${alignr 330}${color FFFFFF}${cpu cpu3}%
${font caviar dreams:bold:size=6}${voffset 3}${goto 20}${color
1B708D}CPU 4${goto 78}${alignr 330}${color FFFFFF}${cpu cpu4}%
${font caviar dreams:bold:size=6}${voffset 3}${goto 20}${color
1B708D}CPU 5${goto 78}${alignr 330}${color FFFFFF}${cpu cpu5}%
${font caviar dreams:bold:size=6}${voffset 3}${goto 20}${color
1B708D}CPU 6${goto 78}${alignr 330}${color FFFFFF}${cpu cpu6}%
${font caviar dreams:bold:size=10}${color 1B708D}${goto 75}${voffset
3}CPU
${font caviar dreams:size=7}${goto 5}${voffset 145}Root:${color
FFFFFF}${alignr 310}${fs_used /} / ${fs_size /}
${font caviar dreams:size=7}${goto 5}${voffset 3}${color
1B708D}Home:${alignr 310}${color FFFFFF}${fs_used /home} / ${fs_size
/home}
${font caviar dreams:bold:size=10}${color 1B708D}${goto 34}${voffset
10}HARD  DRIVE
${goto 5}${voffset 55}
${color FFFFFF}${font caviar dreams:size=8}Uptime: ${uptime_short}
${color FFFFFF}${font caviar dreams:size=8}Processes: ${processes}
${color FFFFFF}${font caviar dreams:size=8}Running:
```

```
${running_processes}
${color 1B708D}${font caviar dreams:size=8}${alignr}${exec cat
/etc/issue.net}  $machine
${color 1B708D}${font caviar dreams:size=8}${alignr}Kernel: ${kernel}


]]
```

Fichier Clock_rings.lua :

clock_rings.lua

```
--[[
Clock Rings by Linux Mint (2011) reEdited by despot77

This script draws percentage meters as rings, and also draws clock
hands if you want! It is fully customisable; all options are described
in the script. This script is based off a combination of my clock.lua
script and my rings.lua script.

IMPORTANT: if you are using the 'cpu' function, it will cause a
segmentation fault if it tries to draw a ring straight away. The if
statement on line 145 uses a delay to make sure that this doesn't
happen. It calculates the length of the delay by the number of updates
since Conky started. Generally, a value of 5s is long enough, so if you
update Conky every 1s, use update_num>5 in that if statement (the
default). If you only update Conky every 2s, you should change it to
update_num>3; conversely if you update Conky every 0.5s, you should use
update_num>10. ALSO, if you change your Conky, is it best to use
"killall conky; conky" to update it, otherwise the update_num will not
be reset and you will get an error.

To call this script in Conky, use the following (assuming that you save
this script to ~/scripts/rings.lua):
    lua_load ~/scripts/clock_rings.lua
    lua_draw_hook_pre clock_rings

Changelog:
+ v1.0 -- Original release (30.09.2009)
   v1.1p -- Jpope edit londonali1010 (05.10.2009)
*v 2011mint -- reEdit despot77 (18.02.2011)
]]

settings_table = {
    {
        -- Edit this table to customise your rings.
        -- You can create more rings simply by adding more elements to
settings_table.
        -- "name" is the type of stat to display; you can choose from
'cpu', 'memperc', 'fs_used_perc', 'battery_used_perc'.
```

```
        name='clock',
        -- "arg" is the argument to the stat type, e.g. if in Conky you
would write ${cpu cpu0}, 'cpu0' would be the argument. If you would not
use an argument in the Conky variable, use ''.
        arg='heure',
        -- "max" is the maximum value of the ring. If the Conky
variable outputs a percentage, use 100.
        max=12,
        -- "bg_colour" is the colour of the base ring.
        bg_colour=0xffffff,
        -- "bg_alpha" is the alpha value of the base ring.
        bg_alpha=0.1,
        -- "fg_colour" is the colour of the indicator part of the ring.
        fg_colour=0x1B708D,
        -- "fg_alpha" is the alpha value of the indicator part of the
ring.
        fg_alpha=0.2,
        -- "x" and "y" are the x and y coordinates of the centre of the
ring, relative to the top left corner of the Conky window.
        x=100, y=150,
        -- "radius" is the radius of the ring.
        radius=50,
        -- "thickness" is the thickness of the ring, centred around the
radius.
        thickness=5,
        -- "start_angle" is the starting angle of the ring, in degrees,
clockwise from top. Value can be either positive or negative.
        start_angle=0,
        -- "end_angle" is the ending angle of the ring, in degrees,
clockwise from top. Value can be either positive or negative, but must
be larger than start_angle.
        end_angle=360
    },
    {
        name='clock',
        arg='minutes',
        max=60,
        bg_colour=0xffffff,
        bg_alpha=0.1,
        fg_colour=0x1B708D,
        fg_alpha=0.4,
        x=100, y=150,
        radius=56,
        thickness=5,
        start_angle=0,
        end_angle=360
    },
    {
        name='clock',
        arg='secondes',
```

```
        max=60,
        bg_colour=0xffffff,
        bg_alpha=0.1,
        fg_colour=0x1B708D,
        fg_alpha=0.6,
        x=100, y=150,
        radius=62,
        thickness=5,
        start_angle=0,
        end_angle=360
    },
    {
        name='time',
        arg='%d',
        max=31,
        bg_colour=0xffffff,
        bg_alpha=0.1,
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=100, y=150,
        radius=70,
        thickness=5,
        start_angle=-90,
        end_angle=90
    },
    {
        name='time',
        arg='%m',
        max=12,
        bg_colour=0xffffff,
        bg_alpha=0.1,
        fg_colour=0x1B708D,
        fg_alpha=1,
        x=100, y=150,
        radius=76,
        thickness=5,
        start_angle=-90,
        end_angle=90
    },
    {
        name='cpu',
        arg='cpu0',
        max=110,
        bg_colour=0xDCDCDC,
        bg_alpha=0.1,
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=100, y=350,
        radius=97,
        thickness=4,
        start_angle=0,
```

```
        end_angle=240
    },
    {
        name='cpu',
        arg='cpu1',
        max=100,
        bg_colour=0xDCDCDC,
        bg_alpha=0.6,
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=100, y=350,
        radius=86,
        thickness=13,
        start_angle=0,
        end_angle=240
    },
    {
        name='cpu',
        arg='cpu2',
        max=100,
        bg_colour=0xDCDCDC,
        bg_alpha=0.5,
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=100, y=350,
        radius=71,
        thickness=12,
        start_angle=0,
        end_angle=240
    },
{
        name='cpu',
        arg='cpu3',
        max=100,
        bg_colour=0xDCDCDC,
        bg_alpha=0.4,
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=100, y=350,
        radius=57,
        thickness=11,
        start_angle=0,
        end_angle=240
    },
    {
        name='cpu',
        arg='cpu4',
        max=100,
        bg_colour=0xDCDCDC,
        bg_alpha=0.3,
```

```
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=100, y=350,
        radius=44,
        thickness=10,
        start_angle=0,
        end_angle=240
    },
{
        name='cpu',
        arg='cpu5',
        max=100,
        bg_colour=0xDCDCDC,
        bg_alpha=0.2,
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=100, y=350,
        radius=32,
        thickness=9,
        start_angle=0,
        end_angle=240
    },
    {
        name='cpu',
        arg='cpu6',
        max=100,
        bg_colour=0xDCDCDC,
        bg_alpha=0.1,
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=100, y=350,
        radius=21,
        thickness=8,
        start_angle=0,
        end_angle=240
    },
    {
        name='fs_used_perc',
        arg='/',
        max=100,
        bg_colour=0xDCDCDC,
        bg_alpha=0.2,
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=110, y=550,
        radius=40,
        thickness=10,
        start_angle=0,
        end_angle=240
    },
    {
```

```
        name='fs_used_perc',
        arg='/home',
        max=100,
        bg_colour=0xDCDCDC,
        bg_alpha=0.2,
        fg_colour=0x1B708D,
        fg_alpha=0.8,
        x=110, y=550,
        radius=28,
        thickness=10,
        start_angle=0,
        end_angle=240
    },
}

-- Use these settings to define the origin and extent of your clock.

clock_r=65

-- "clock_x" and "clock_y" are the coordinates of the centre of the
clock, in pixels, from the top left of the Conky window.

clock_x=100
clock_y=150

show_seconds=true

require 'cairo'

function rgb_to_r_g_b(colour,alpha)
    return ((colour / 0x10000) % 0x100) / 255., ((colour / 0x100) %
0x100) / 255., (colour % 0x100) / 255., alpha
end

function draw_ring(cr,t,pt)
    local w,h=conky_window.width,conky_window.height

    local
xc,yc,ring_r,ring_w,sa,ea=pt['x'],pt['y'],pt['radius'],pt['thickness'],
pt['start_angle'],pt['end_angle']
    local bgc, bga, fgc, fga=pt['bg_colour'], pt['bg_alpha'],
pt['fg_colour'], pt['fg_alpha']

    local angle_0=sa*(2*math.pi/360)-math.pi/2
    local angle_f=ea*(2*math.pi/360)-math.pi/2
    local t_arc=t*(angle_f-angle_0)

    -- Draw background ring

    cairo_arc(cr,xc,yc,ring_r,angle_0,angle_f)
```

```
    cairo_set_source_rgba(cr,rgb_to_r_g_b(bgc,bga))
    cairo_set_line_width(cr,ring_w)
    cairo_stroke(cr)

    -- Draw indicator ring

    cairo_arc(cr,xc,yc,ring_r,angle_0,angle_0+t_arc)
    cairo_set_source_rgba(cr,rgb_to_r_g_b(fgc,fga))
    cairo_stroke(cr)
end

function draw_clock_hands(cr,xc,yc)
    local secs,mins,hours,secs_arc,mins_arc,hours_arc
    local xh,yh,xm,ym,xs,ys

    secs=os.date("%S")
    mins=os.date("%M")
    hours=os.date("%I")

    secs_arc=(2*math.pi/60)*secs
    mins_arc=(2*math.pi/60)*mins+secs_arc/60
    hours_arc=(2*math.pi/12)*hours+mins_arc/12

    -- Draw hour hand

    xh=xc+0.7*clock_r*math.sin(hours_arc)
    yh=yc-0.7*clock_r*math.cos(hours_arc)
    cairo_move_to(cr,xc,yc)
    cairo_line_to(cr,xh,yh)

    cairo_set_line_cap(cr,CAIRO_LINE_CAP_ROUND)
    cairo_set_line_width(cr,5)
    cairo_set_source_rgba(cr,1.0,1.0,1.0,1.0)
    cairo_stroke(cr)

    -- Draw minute hand

    xm=xc+0.85*clock_r*math.sin(mins_arc)
    ym=yc-0.85*clock_r*math.cos(mins_arc)
    cairo_move_to(cr,xc,yc)
    cairo_line_to(cr,xm,ym)

    cairo_set_line_width(cr,3)
    cairo_stroke(cr)

    -- Draw seconds hand

    if show_seconds then
        xs=xc+clock_r*math.sin(secs_arc)
        ys=yc-clock_r*math.cos(secs_arc)
        cairo_move_to(cr,xc,yc)
```

```
        cairo_line_to(cr,xs,ys)

        cairo_set_line_width(cr,1)
        cairo_stroke(cr)
    end
end

function conky_clock_rings()
    local function setup_rings(cr,pt)
        local secs, mins, hours, mins_secs, hours_mins
        local str=''
        local value=0

        if pt['name']=='clock' then
            secs=os.date("%S")
            mins=os.date("%M")
            hours=os.date("%I")

            mins_secs=mins+secs/60
            hours_mins=hours+mins/60
            if hours_mins >= 12 then hours_mins=hours_mins-12 end
            if pt['arg']=="heure" then
            str=hours_mins
            elseif pt['arg']=="minutes" then
            str=mins_secs
            else
            str=secs
            end
        else
            str=string.format('${%s %s}',pt['name'],pt['arg'])
            str=conky_parse(str)
        end

        value=tonumber(str)

        if value==nil then -- Gestion du problème de séparateur
décimale
            str=conky_parse(str):gsub("%.",",")
            value=tonumber(str)
        end

        if value == nil then value = 0 end
        pct=value/pt['max']

        draw_ring(cr,pct,pt)
    end

    -- Check that Conky has been running for at least 5s

    if conky_window==nil then return end
```

```
    local
cs=cairo_xlib_surface_create(conky_window.display,conky_window.drawable
,conky_window.visual, conky_window.width,conky_window.height)

    local cr=cairo_create(cs)

    local updates=conky_parse('${updates}')
    update_num=tonumber(updates)

    if update_num>5 then
        for i in pairs(settings_table) do
            setup_rings(cr,settings_table[i])
        end
    end

    draw_clock_hands(cr,clock_x,clock_y)
end
```

Logo Debian au centre de la pendule (debian-logo.png):