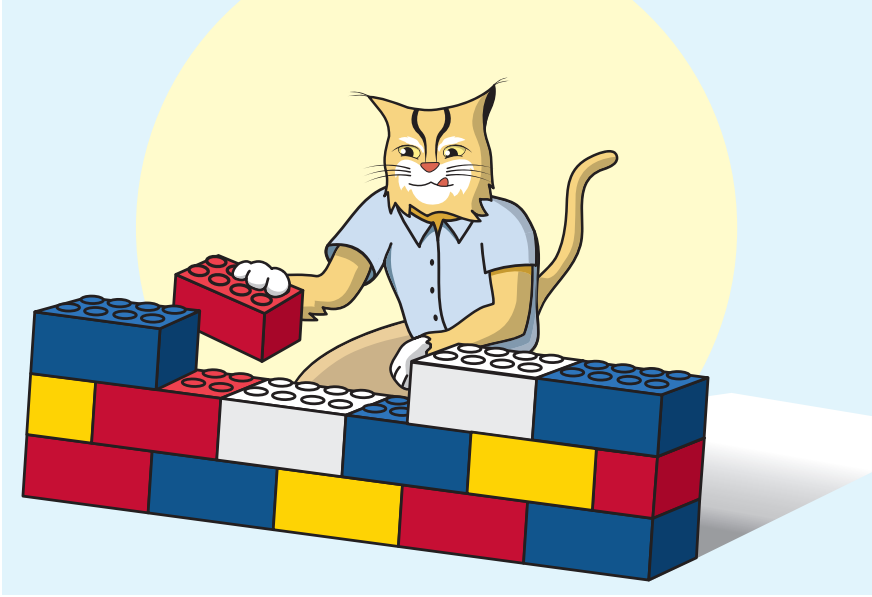
# King of the Desktop



Turn your desktop into a dashboard that displays system monitoring data, weather updates, news headlines, and more with Conky. BY DMITRI POPOV

When it comes to system monitoring tools, Conky [1] rules the roost supremely. This lightweight utility can help you keep an eye on virtually any aspect of your system, and it offers a long list of options for you to tweak. But displaying various information about your system is only one of Conky's many talents: You can use this nifty tool to pull headlines from RSS feeds, check email, view the weather forecast, and even execute commands and display output right on your desktop. In other words, Conky lets you turn the desktop into a powerful dashboard that feeds you all the important information.

Configuring Conky can be a daunting proposition, but despair not: I'll demonstrate how to tweak Conky's settings and put this powerful tool to some clever uses.

## Install Conky

Because Conky is available in Ubuntu software repositories, installing it is a matter of running the *sudo apt-get install conky* command. Besides the core package, you might want to install a few third-party additions that extend Conky's functionality.

To do this, you have to add the *conkyhardcore* personal package archive (PPA) using the following command:

```
sudo apt-add-repository ➥
  ppa:conkyhardcore/ppa && ➥
  sudo apt-get update
```

## Getting a Taste of Conky

Before you get your hands dirty tweaking Conky's configuration file, you might want to get a taste of Conky's capabilities using one of many ready-to-use solutions like Conky Ubuntu Lucid Theme (Figure 1) [2]. Grab the latest version of the theme, unpack the downloaded archive and move the resulting *.conkytheme* folder to your home directory.

Launch the terminal and run the following command:

```
conky -c ~/.conkytheme/conkyrc
```

You should then see the Conky panel in all its beauty at the bottom of the screen.

## Configuring Conky

By default, Conky's configuration options are stored in the *.conkyrc* file, which is split into two parts: the first controls Conky's appearance, and the second specifies which parameters to monitor and how to display them. To learn the ropes, I'll show you how to create a configuration file from scratch. So, fire up your favorite text editor to get ready to go.

To begin, you must specify the update interval, or how often Conky "collects" the monitored data. This is done by specifying the *update_interval* option in seconds, for example *update_interval 1.0* or *update_interval 5.0*. You must also set the *total_run_times* option to *0* to let Conky run until you close it.

By default, Conky uses a monospace font, but you can instruct it to use any font installed on your system by enabling the *use_xft* option and specifying a font in the *xftfont* setting. For example, if you want Conky to use Droid Sans Mono 9pt font, these options should be:

```
use_xft yes
xftfont Droid Sans Mono:size=9
override_utf8_locale yes
```

The last parameter forces Conky to use UTF8 encoding when *use_xft yes* is enabled.

Quite often, the Conky window might flicker during refreshing, which can be very annoying. Conky has two options to help you to combat that problem. The *double_buffer* option lets Conky use the X double-buffer extension that can eliminate flickering. If enabling *double_buffer* doesn't help, you can use the *own_window* option that forces Conky to run in a separate window. To be on the safe side, you might want to enable both options:

```
double_buffer yes
own_window yes
```

The next step is to specify the window settings, including its width, alignment, and default color. In the example below, Conky renders a transparent 300-pixel-wide window superimposed on the desktop in the top right corner of the screen:

```
maximum_width 300
own_window_transparent yes
own_window_type override
alignment top_right
```

By default, you can place the Conky window at any of the four corners of the screen using the *alignment top_left*, *alignment top_right*, *alignment bottom_left*, and *alignment bottom_right* options. But what if you want to move the window slightly to the left or a little bit down? To do this, you can use the *gap_x* and *gap_y* options, which specify the distance between the borders of the screen and the window:

```
gap_x 10
gap_y 35
```

If you don't want text in the Conky window to appear in caps, use the *uppercase no* option. Conky offers many more configuration options for you to tweak, but these will get you started.

With the basic display options in place, you can start adding and configuring the system parameters you want Conky to monitor. This is done using so-called variables. For example, to display the kernel version of your system, you can use the *kernel* variable. Note that each variable is preceded by the *$* sign. Conky also includes a few variables that allow you to control the appearance of the output. The *color* variable allows you to specify text color, whereas the *alignr* variable allows you to right-justify text. To see how this works in practice, take a look at the following example:

```
${color lightgrey}Uptime:$color ➥
  $uptime $alignr${color lightgrey}➥
  Load:$color $loadavg
```

The *${color lightgrey}* variable sets the text to light gray. To switch to the default color (white), use the *color* variable without a parameter.

Conky sports several graphing variables that you can use to visualize monitored data. The *cpugraph* variable, for example, displays CPU usage as a graph. When used without any parameters, the graph is shown using the default solid color, but you can use Hex color values to specify gradients, which make the graph look much better:

```
${cpugraph 000000 ffffff}
```

To display the memory-related data, you can use the *memmax* (the total amount of RAM), *mem* (memory in use), and *memperc* (percentage of memory in use) variables:

```
${color lightgrey}RAM usage:$color ➥
  $mem/$memmax - $memperc%
```

When it comes to monitoring disk usage, you have a few variables from which to choose, including *fs_free* (free space), *fs_size* (total size), and *fs_bar* (space used bar):

```
${color grey}Disk usage: ➥
  $color${fs_free /} of ➥
   ${fs_size /}
${fs_bar 6 /}
```

Conky also includes several variables that allow you to monitor wireless interfaces, including *wireless_essid* (returns the ESSID name of the access point), *wireless_link_qual* (returns wireless link quality), and *wireless_link_bar* (returns the wireless link quality displayed as a bar).

To obtain the name of the wireless interface you will want to monitor with these variables, use the *ifconfig* command, which returns a list of available network interfaces. Then, you can use the variables to display the desired information about a specific wireless interface as follows::

```
${color #ffcb48}Wi-Fi ${hr 1}
${color lightgrey}ESSID: ➥
  $color${wireless_essid wlan0}
${color lightgrey}Wireless signal: ➥
  $color${wireless_link_qual wlan0}%
${wireless_link_bar wlan0}
```

If you want to keep an eye on the download and upload speed, you can use the *downspeedf* and *upspeedf* variables (or *downspeed* and *upspeed* variables if you don't want to display the trailing decimal):

```
${color lightgrey}Download speed: ➥
  $color${downspeedf wlan0} Kb/sec
```

Alternatively, you can use the *downspeedgraph* and *upspeedgraph* variables to display the download and upload speeds as graphs. And if you want to monitor the total amount of downloaded and uploaded data, you can use the *totaldown* and *totalup* variables:

```
${color red}Down: ➥
  $color${totaldown wlan0} ➥
  $alignr ${color green}Up: ➥
   $color${totalup wlan0}
```

This can come in particularly handy if you are using a connection with a download cap.

## Clever Tricks

Conky includes several conditional variables that give you even more flexibility. For example, if your machine has a wired and wireless interface, you might want to monitor both of them. But usually you use only one interface at a time, so you don't need Conky to display information about the inactive interface. This is where the *if_existing*, *else*, and *endif* variables come into the picture. Using them, you can specify a condition that displays information about a specific interface only when it's active; otherwise, the condition either hides the unnecessary data or displays a message like "No Network Connection":

```
${if_existing /proc/net/route eth0}➥
  ${color #ffcb48}Ethernet ${hr 1}
${color lightgrey}IP address: ➥
  $alignr$color${addr eth0}${else}➥
  No Network Connection${endif}
```

Besides the system-related variables, Conky has several other variables that can be used to pull and to process data from other sources. For example, using the *rss* variable, you can use Conky to go out and get the very latest headlines from your favorite RSS feed:

```
${rss http://www.linux-magazine.com/➥
  rss/feed/productivity_sauce 10 ➥
  item_titles 5}
```

In the example above, Conky fetches the five most recent headlines from the Productivity Sauce blog's RSS feed every 10 minutes. Because Identi.ca and Twitter publish updates via RSS, you can use the *rss* variable to keep an eye on updates from people you follow:

```
${rss http://identi.ca/api/statuses/➥
  friends_timeline/dmpop.rss ➥
  15 item_titles 7}
```

Perhaps the most versatile variable in Conky's arsenal is *execi*. It allows you to run any shell command and display the output in the Conky window. You can put this functionality to many nifty uses. For example, you can create a plain-text file for your to-do list and then use the following command to display it in the Conky window:

```
${execi 30 cat ~/TODO.txt | fold -w40}
```

The *execi* variable can do more advanced tricks too, like displaying the number of incoming email messages in your inbox. For that, you can use the *conkyemail* package from the *conkyhardcore* repository you installed earlier. Just install the package with *sudo apt-get install conkyemail*. Then, issue the *conkyEmail --help* command to view a list of all available options. You can use these options to create an appropriate *execi* command. If you use Gmail, you can use the command below to poll your Gmail account (replace the *USER* and *PASSWORD* values with your actual Gmail user name and password):

```
${color #ffcb48}${➥
 execi 30 conkyEmail ➥
 --servertype=IMAP ➥
  --servername=imap.gmail.com ➥
  --port=993 --ssl --username=USER ➥
  --password=PASSWORD} new emails
```

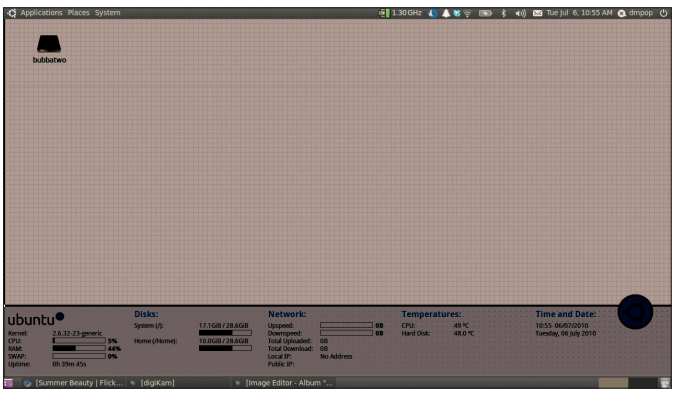



**Figure 1: Conky Lucid Theme in all its beauty.**

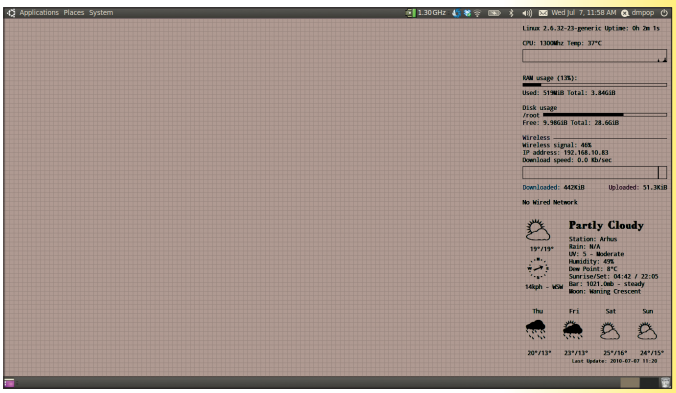



**Figure 2: Here is what Conky looks like on my notebook.**

The *conkyhardcore* repository also contains a package called *conkyforecast*, which can pull weather forecasts and present them on your desktop. Configuring *conkyforecast* requires some work, but the results are worth it.

To start, install the *conkyforecast* package:

```
sudo apt-get install conkyforecast
```

To pull weather data, you have to obtain a Partner ID and License Key from The Weather Channel service [3] and follow the instructions for an account. Then, copy the config file into your home directory,

```
cp /usr/share/conkyforecast/➥
  conkyForecast.config ➥
  ~/.conkyForecast.config
```

open the copied file in your text editor, and enter the Partner ID and License Key:

```
XOAP_PARTNER_ID = Your Partner ID
XOAP_LICENCE_KEY = Your License Key
```

Next, you have to find the Location ID of your city. To do this, query Weather.com for the city code using the name of the city – for example:

```
http://xoap.weather.com/search/➥
  search?where=NEW YORK
```

Make a note of the returned value (in this case, it is *USNY0996*). Instead of manually configuring the presentation layout, you can use the sample template. To copy the template to your home directory, use the following command:

```
cp /usr/share/conkyforecast/example/➥
  conkyForecast.template ➥
  ~/conkyForecast.template
```

Then, add this command to the *.conkyrc* file as follows:

```
${execpi 1800 conkyForecast ➥
  --location= USNY0996 ➥
  --template=~/conkyForecast.template}
```

Now when you launch Conky, you should be able to see the weather forecast in all its beauty (Figure 2).

Although Conky lets you display all kinds of useful information, your screen's height limits how much data you can show on your desktop. But there is a workaround to this problem: You can launch two separate instances of Conky, each using its own configuration file. You can configure the *.conkyrc* file to place one Conky window in the top right corner of the screen and a *.conkyrc2* file to display another window in the top left corner. Then, you can use the following command to launch two Conky instances:

```
(conky &);(conky -c .conkyrc2 &)
```

## Final Word

To wrap up, you can see the *.conkyrc* file that I use in Listing 1 [4]. Feel free to use this example as a starting point for creating your own Conky configuration. ■

### Info

[1] Conky: http://*conky.sourceforge.net*

[2] Conky Ubuntu Lucid Theme: *http://bit.ly/bJvQrT*

[3] Weather data feed: *http://www.weather.com/services/xmloap.html*

[4] Code for this article: *http://www.ubuntu-user.com/Online/Article-Code*

### Listing 1: .conkyrc Sample

```
02
03 update_interval 1.0
04 double_buffer yes
05 background no
06
07 use_xft yes
08 xftfont Droid Sans Mono:size=9
09 override_utf8_locale yes
10 text_buffer_size 2048
11
12 maximum_width 300
13 own_window yes
14 own_window_transparent yes
15 own_window_type override
16 alignment top_right
17
18 gap_x 10
19 gap_y 35
20
21 TEXT
22 $font$color$sysname $kernel ${color lightgrey}Uptime:
       $color $uptime
23
24 ${color lightgrey}CPU: $color${freq}Mhz
   ${color lightgrey}Temp:$color $acpitemp°C
25 $color${cpugraph}
26
27 ${color lightgrey}RAM usage $color($memperc%):
28 ${membar}
29 ${color lightgrey}Used: $color$mem
   ${color lightgrey}Total: $color$memmax
30
31 ${color grey}Disk usage
32 /root $color${fs_bar 6 /}
33 ${color grey}Free: $color${fs_free /}
   ${color grey}Total: $color${fs_size /}
34
35 ${if_existing /proc/net/route wlan0}
   ${color #ffcb48}Wireless ${hr 1}
36 ${color lightgrey}Wireless signal:
   $color${wireless_link_qual wlan0}%
37 ${color lightgrey}IP address: $color${addr wlan0}
38 ${color lightgrey}Download speed:
   $color${downspeedf wlan0} Kb/sec
39 ${downspeedgraph wlan0}
40 ${color red}Downloaded: $color${totaldown wlan0} $alignr
   ${color green}Uploaded: $color${totalup wlan0}
   ${else}No Wireless Network${endif}
41
42 ${if_existing /proc/net/route eth0}
   ${color #ffcb48}Ethernet ${hr 1}
43 ${color lightgrey}IP address: $color${addr eth0}
44 ${color lightgrey}Download speed:
   $color${downspeedf eth0} Kb/sec
45 ${downspeedgraph eth0}
46 ${color red}Downloaded: $color${totaldown eth0} $alignr
   ${color green}Uploaded: $color${totalup eth0}
   ${else}No Wired Network${endif}
47
48 ${execpi 1800 conkyForecast --location=DAXX0003
     --template=~/conkyForecast.template}
```