Conky, der ukltimative Systemmonitor – Teil II

von Stefan Fuhrmann (searchOne)

Im ersten Teil der Conky-Serie (MagDriva 2.2009) wurden die *Basics* im Umgang mit Conky besprochen, die wir im zweiten Teil nun ausbauen und vertiefen werden. Anhand von umfangreicheren Beispielen steigen wir noch tiefer in die Materie ein. Mit diesen Grundlagen habt ihr genügend Wissen gesammelt um euren eigenen Conky problemlos zu erstellen.

Noch ein Wort für die etwas Erfahreneren *Conky-User:* In diesem HowTo werden nur die (meiner Meinung nach) wichtigsten Optionen kurz besprochen, anhand derer die Funktionsweise von Conky am besten erklärt werden kann. Weitere Infos stehen im *„Mandriva Wiki“*, auf der *„Conky HomePage“* oder bei *„Conky Hardcore“*. Viel Spaß dabei!

Conky und Autostart

Eine Möglichkeit ist, Conky in die Autostart-Datei einzutragen (die Autostart-Dateien sind bei Gnome unter /home/user/.config/autostart zu finden) oder z.B. über das KDE Kontrollzentrum

> Die Arbeitsumgebung konfigurieren > Erweitert > Autostart.

Eine zweite Möglichkeit ist, conky *von Hand* zu starten

```
> Alt+F2 > conky > [ENTER]
```

Danach sollte der fertige Desktop über die Einstellung *Vorige Sitzung wiederherstellen* gespeichert werden, zu finden unter:

> Die Arbeitsumgebung konfigurieren > Erweitert > Sitzungsverwaltung > Bei der Anmeldung > Vorige Sitzung wiederherstellen.

Damit ist ein Eintrag in der Autostart nicht mehr nötig!

ACHTUNG: Sollte Conky in der Autostart stehen und zusätzlich der Desktop mit der Einstellung *Vorige Sitzung wiederherstellen*, dann wird Conky mehrfach gestartet!

Der Trick mit den Linien

Um eine horizontale Linie in Conky zu zeichnen, benötigen wir die *$hr* Variable. Damit können wir eine horizontale Linie von einem beliebigen Anfang bis zum gegenüberliegenden Ende des jeweiligen Conky-Fensters zeichnen. Beispiel:

```
${color red}${hr 2}
```

Hier erzeugen wir eine rote Linie 2 Pixel hoch bis zum Ende des Fensters. Beispiel:

```
${goto 20}${color red}${hr 2}
```

Das gleiche wie oben, nur ist der Anfang nun 20 Pixel vom linken Rand (dank der *$goto* Variablen) des Fens-

ters entfernt! Wenn wir nun die Linie auch auf der rechten Seite begrenzen wollen, gibt es zwei Möglichkeiten. Beispiel:

```
${goto 20}${color red}${alignr 120}${hr 2}
```

Wir müssen einfach ein *$alignr* mit der gewünschten Länge vor dem *$hr* einfügen! Somit wird die Linie auch auf der rechten Seite begrenzt.

Die zweite Möglichkeit ist ein kleiner Trick (danke an „Bruce“ von – Conky Hardcore):

```
${goto 20}${color red}${cpubar cpu9 2,120}
```

Gleiches Ergebnis wie oben, nur haben wir hier dank der Funktion *bar* in Conky und einer „nicht vorhandenen CPU“ (sehr wichtig) ebenfalls eine Linie in der gewünschten Länge erzeugt.

Conky Transparent

Kommen wir nun zu einem der wichtigsten Themen (zumindest in den Foren rund um den Globus!) unter Conky, nämlich der transparenten Darstellung des selbigen. In den meisten Fällen gibt es keinerlei Probleme, so z.B. unter GNOME, Xfce oder OpenBox. Jedoch bei KDE sind einige Besonderheiten zu beachten, die wir hier kurz Erläutern werden.

Unter **KDE 3.x** muss die Option *Programme im Arbeitsflächenfenster unterstützen* aktiviert werden, die man im Kontrollzentrum (Die Arbeitsumgebung konfigurieren) unter *Arbeitsfläche > Verhalten* im Reiter *Allgemein* findet. Bei Verwendung der Pseudo-Transparenz wird jedoch ein schwarzer Hintergrund angezeigt, da KDE das Hintergrundbild nicht in das Root-Fenster schreibt, sondern eine Ebene darüber. Das Programm *qiv* schafft Abhilfe.

Folgendes Paket muss also zusätzlich installiert werden (siehe Teil 1, Installation von Conky): `qiv`

Nun kann das Hintergrundbild in der Konsole gesetzt werden:

```
qiv --root /pfad/zum/bild.png
```

Damit dies bei jedem Conky-Start gemacht wird, trägt man folgende Zeile in die .conkyrc:

```
${exec /usr/bin/qiv --root /pfad/zum/bild.png}
```

Für KDE 4.x müssen wir ein anderes Programm installieren: `feh`

Danach wechseln wir in das KDE Kontrollzentrum > *Die Arbeitsumgebung konfigurieren > Erweitert > Autostart*. Dort machen wir einen Eintrag unter > *Einrichtungs-Datei(.desktop)* über den Reiter > *Eigenschaften > Programm > Befehl* fügen wir den Eintrag

```
feh --bg-scale /home/pfad/zum/Bild.png
```

Mit `Bild` ist hier das aktuelle Hintergrundbild gemeint. Nun wird bei jedem Start *feh* automatisch geladen und das Hintergrundbild wird zusätzlich *hinter* conky eingefügt, somit ist die Transparenz komplett.

Ab **KDE 4.x** kann *feh* auch direkt aus der *~/.conkyrc* gestartet werden (Befehl in einer Zeile!):

```
${texeci 3600 feh --bg-scale "`grep 'wallpaper=' ~/.kde4/share/config/plasma-appletsrc | tail -n1 | tail –bytes=+11`"}
```

oder (Befehl in einer Zeile!):

```
${texeci 3600 feh --bg-scale
"`grep 'wallpaper=' ~/.kde4/sha-
re/config/ plasma-appletsrc |
tail –bytes=+11`"}
```

In diesem Fall sucht sich das Programm *feh* den Name des Wallpapers aus den KDE4-Dateien selbst heraus und trägt ihn in die Zeile ein!

Achtung: Scheinbar gibt es bei KDE noch einige kleine Unterschiede, denn bei einigen älteren Versionen sollte diese Zeile eingefügt werden (Befehl in einer Zeile!):

```
${texeci 3600 feh --bg-scale
"`grep 'wallpaper=' ~/.kde/sha-
re/config/plasma-appletsrc |
tail -n1 | tail --bytes=+11`"}
```

KDE 4.3.2

Vor wenigen Wochen wurde die neue *Mandriva Version 2010.0 Adelie* veröffentlicht! Im gleichen Zuge wurde in der Mandriva Umgebung auch KDE 4.3.2 eingeführt, was einige Änderungen nach sich zog.

Die auffälligste schlägt sich gleich mal in den Einstellung für die Transparenz von Conky nieder! Der Datei-name, aus der sich *feh* das passende Hintergrundbild besorgt, die *plasma-appletsrc*, hat sich geändert in *plasma-desktop-appletsrc*.

Das bedeutet also, die Einstellungen für *feh* müssen angepasst werden in (Befehl in einer Zeile!):

```
${texeci 1000 feh --bg-scale
"`grep 'wallpaper=' ~/.kde4/sha-
re/config/plasma-desktop-app-
letsrc|head -n1|tail
--bytes=+11`"}
```

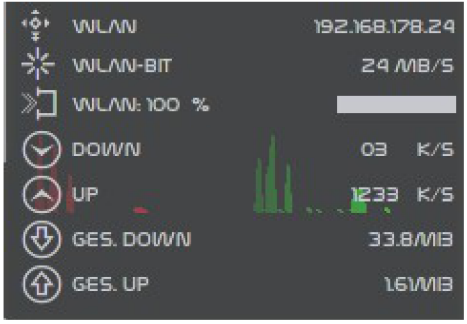

Abbildung 1: Graphen- und Baranzeigen

Diagramme und Systemdaten

Conky kann verschiedene Diagramme in Form von Leisten und Graphen darstellen (auch mit unterschiedlichen Farben, Siehe Bild 1). Als Beispiel sehen wir hier die Auslastung der Netzwerkkarte *wlan0*. Bei einem Graphen, hier *$downspeedgraph*, muss das Interface, in diesem Fall *wlan0*, die Höhe und Breite *40,110* und die Gradienten-Farben *330000 ff0000*. Optional kann auch eine andere Skalierung in Zahlenwerten angegeben werden.

```
${downspeedgraph wlan0 40,11
330000 ff0000}

${upspeedgraph wlan0 40,110
003300 00ff00}
```

In diesem Beispiel wurde zusätzlich zu dem Diagramm noch eine normale Anzeige in Schriftform vor das Diagramm gelegt. Zu realisieren ist das über die *$voffset* Variable (siehe Teil 1, Text Formatieren)mit positiven und negativen Wertangaben wird die Anzeige soweit verschoben bis sie hinter die Schrift verrutscht.

Das nächste Beispiel ist eine Leiste als WLAN Empfangssignal in Prozent-

werten (siehe Bild 1). Einmal die Varibale *$wireless_link_bar*, die eine Leiste erzeugt, danach Höhe und Breite *8,60* und am Ende dann das Interface *wlan0* (Befehl in einer Zeile!):

```
WLAN: ${wireless_link_qual_perc
wlan0}% ${alignr} $
{wireless_link_bar 8,60 wlan0}
```

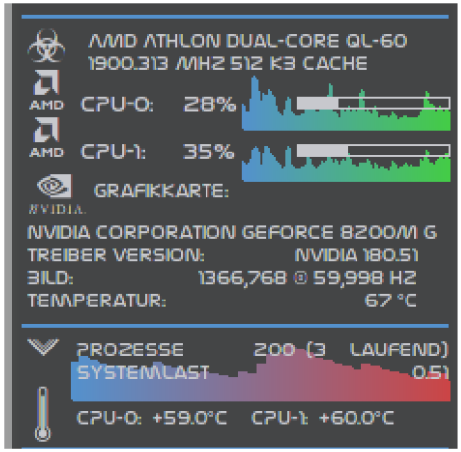

Abbildung 2: Diagramme

Aufbau und Schreibweise sind bei allen Graphen, Diagrammen und Baranzeigen gleich. Es wechselt nur das Interface, z.B. *wlan0*, *bat1*, *cpu*, *cpu1*, *eth2* usw. (siehe Bild 2).

Noch eine Anmerkung zu den Bildern, denn es taucht immer wieder die Frage auf nach den Symbolen vor der Beschriftung.

Das sind auch nur *Fonts*! Ein sehr guter Link dazu ist

http://www.dafont.com

Dort gibt es Unmengen an Fonts in allen Arten und Formen zu finden. Nach der Installation der Fonts im System wird er folgendermaßen in Conky eingebunden:

```
${font StyleBats:size=16}P$
{font}
```

Mit der Variable *$font* wird der Font aufgerufen, dahinter kann dann Optional ein *size=X* folgen mit der dann die Schriftgröße angepasst wird. Nun noch der Buchstabe, der Stellvertretend für das Symbol in der *~/.conkyrc* erscheint, hier eben das *P*. Dahinter dann noch ein einfaches *${font}*, mit dem wir Conky mitteilen das die weitere Anzeige wieder über den normalen Font erfolgen soll!

Die CPU

Kommen wir nun zum Herz eines jeden PC's, der CPU! Zuerst beschaffen wir uns einen Überblick über die möglichen Informationen, die uns das System anbietet. Das passiert in der Konsole mit folgendem Befehl:

```
cat /proc/cpuinfo
```

Hiermit bekommen wir eine Übersicht angezeigt welche Werte wir aus der CPU auslesen können. In der ersten Spalte steht die Variable die uns den Wert anzeigt, der uns Interessiert. Dahinter folgt die Ausgabe, getrennt mit einem Doppelpunkt. Das ganze wird dann mit *grep* kombiniert und hat z.B. folgendes Format (Befehl in einer Zeile!):

```
${execi 99999 cat /proc/cpuinfo
| grep "model name" -m1 | cut
-d":" -f2 | cut -d" " -f2-}
```

Der erste Teil ruft das Programm *cat /proc/cpuinfo* auf, gefolgt von *grep* der nach der gewünschten Variablen sucht, die hier angegeben ist: *grep "model name"*. Danach werden nur noch der Doppelpunkt und das Leerzeichen entfernt und wir bekommen den Modellnamen der CPU als

Ausgabe präsentiert. Zwei weitere Beispiele:

```
${execi 99999 cat /proc/cpuinfo | grep "cpu MHz" -m1 | cut -d":" -f2 |cut -d" " -f2-} Mhz
```

und

```
${execi 99999 cat /proc/cpuinfo | grep "cache size" -m1 | cut -d":" -f2 |cut -d" " -f2-} Cache
```

Im ersten lassen wir uns die *Mhz* der CPU anzeigen, darunter den *cache size*. Zusätzlich gibt es in Conky auch noch weitere Variablen für die *CPU*, z.B. *$cpugraph* oder *$cpubar* zur Graphischen Darstellung der Werte:

```
${cpugraph cpu1 40, 110 1E90FF 00FF00} ${cpubar cpu0 6,80}
```

Oder aber eine einfache Prozentanzeige mit *${cpu}*:

```
CPU-0:${goto 90}${cpu cpu0}%
```

Temperaturanzeige

Ein weiteres Thema ist die Temperaturanzeige. Beginnen möchte ich mit den Werten der verbauten Festplatten. Zuerst muss über die Paketverwaltung (MCC) das Programm *hddtemp* installiert werden. Danach kann mit einem einfachen Befehl in der Konsole geprüft werden, ob es korrekt funktioniert:

```
hddtemp /dev/sda
```

Gegebenenfalls hier die Festplattenbezeichnung anpassen. Sollte alles korrekt sein, bekommt man folgende Ausgabe:

```
hddtemp /dev/sda
/dev/sda: WDC WD1600BEVT-60ZCT1: 45°C
```

In unserem Beispiel bekommen wir die Bezeichnung der Platte sowie am Ende die aktuelle Temperatur angezeigt. Hier können wir hier schon unsere zukünftige Ausgabe für Conky anpassen. Das machen wir mit folgendem Befehl:

```
hddtemp /dev/sda | cut -c 36-40
```

Die Ausgabe von *hddtemp* wird über eine Pipe (das ist der | Strich) an *cut* geleitet, der dann die gewünschte Zeichenfolge an der Position *36-40* herausschneidet.

Das hätte hier folgende Ausgabe zur Folge:

```
hddtemp /dev/sda | cut -c 36-40
°C
```

Wie man gut sehen kann, wird die Ausgabe an der falschen Stelle beschnitten. Also muss der hintere Wert von *cut* angepasst werden.

```
hddtemp /dev/sda | cut -c 34-37
45°C
```

Jetzt stimmt die Ausgabe. Wer noch zusätzlich den Namen der Festplatte oder auch die Bezeichnung möchte, muss den Wert entsprechend anpassen. Nun wird alles noch in die *~/.conkyrc* gepackt mit:

```
${execi 10 hddtemp /dev/sda | cut -c34-37}
```

lmsensors

Sollte das Paket *lm_sensors* Installiert sein, kann auch die Temperatur der einzelnen *CPU's* ausgelesen werden. Auf der Konsole kann das mit einem einfachen

```
sensors
```

überprüft werden.

Hier ein Beispiel für Conky (Befehl in einer Zeile!):

```
CPU-0:  ${execi 10 sensors | sed -n '3p' | sed 's/temp1:[ ]*//' | cut -d" " -f1}
```

Zudem können noch diverse andere Temperaturen erfasst werden (z.B. Festplatten, Grafikkarten usw.), falls der PC und die Software diese Funktionen unterstützten!

Anzeige von Systemprozessen

Mit wenigen Zeilen in der *~/.conkyrc* kann man sich die aktuellen System-Prozesse Anzeigen lassen:

```
TOP 5 PROCESSES
NAME PID CPU MEM
${color orange}1. ${top name 1}${top pid 1} ${top cpu 1} ${top mem 1}$color
2. ${top name 2}${top pid 2} ${top cpu 2} ${top mem 2}
3. ${top name 3}${top pid 3} ${top cpu 3} ${top mem 3}
4. ${top name 4}${top pid 4} ${top cpu 4} ${top mem 4}
5. ${top name 5}${top pid 5} ${top cpu 5} ${top mem 5}
```

Die ersten beiden Zeilen sind nur Überschriften. Die dritte Zeile ist die erste Anzeige des Top-Processes, der hier immer in Orange eingefärbt wird. Darunter dann die Prozesse 2-5. Angezeigt werden der Name *${top name 1}* die *PID ${top pid 1}* die Belastung der *CPU ${top cpu 1}* und den Verbrauch an *RAM ${top mem 1}*.

Insgesamt können bis zu 10 Prozesse abgebildet werden mit den folgenden Variablen (Format wie oben im Beispiel, also *${top mem}*):

name, pid, cpu, mem, mem_res, mem_vsize, und *time*.

Ausblick auf Teil III

Das war es für dieses Mal. Im dritten Teil werden wir uns noch die Schleifen-Funktionen von Conky ansehen, und dann mit einem Ausblick auf Conky 1.7.2 unser kleines Conky - HowTo beenden.