Temperaturüberwachung mit Linux (Teil 2)

by Stefan Blechschmidt
<sb(at)sbsbavaria.de>

About the author:

Als gelernter Elektriker hat man mich 1990 vor einen CAD Arbeitsplatz gesetzt, um eine Schaltanlage zu planen. Anscheinend habe ich mich damals mit einem noch unbekannten Virus infiziert, und das ist gut so.

Abstract:

In der Ausgabe November 2003 Temperaturüberwachung mit Linux habe ich eine Schaltung beschrieben, mit der man Temperaturwerte mittels Linux erfassen kann. Um diese Temperaturwerte nun ordentlich auswerten zu können, sollte man diese in einer Datenbank speichern.

Um den ganzen Beitrag abzurunden, werden wir die Daten noch über ein Web−Interface grafisch darstellen.

Voraussetzung

Auf deinem Rechner sollten schon einige Anwendungen installiert sein und natürlich funktionieren.

- Perl
- Apache
- MySQL
- und ein paar Perlmodule, die uns das Erstellen der Programme erleichtern, aber dazu später mehr.

Wie man sehen kann, richtet sich der Artikel an den schon etwas erfahrenen Linuxuser. Für alle anderen ist es sicher ein guter Einstieg ;−)

Einrichten der Datenbank

Unter MySQL bildet das gleichnamige Programm *mySQL* die Schnittstelle zur Datenbank. Mit dem Befehl `mysql -u root -p mysql` kommen wir in den MySQL Monitor.

Mit dem Schalter `-u` wird der Benutzer angegeben. Mit dem Schalter `-p` wird das Passwort interaktiv abgefragt, am Schluss wird noch die Datenbank angegeben, die verwendet werden soll. Wobei wir, in diesem Beispiel, die Verwaltungsdatenbank von MySQL ausgewählt haben.

Du erhältst jetzt den Prompt `mysql >` bei dem die SQL Befehle eingegeben werden. Als erstes wollen wir uns ansehen, welche Tabellen in der Datenbank enthalten sind. Mit dem Befehl `show tables;` bekommen wir die Übersicht.

```
mysql> show tables;
+-----------------+
| Tables_in_mysql |
+-----------------+
| columns_priv    |
| db              |
| func            |
| host            |
| tables_priv     |
| user            |
+-----------------+
6 rows in set (0.00 sec)
```

Jetzt geht es daran, die Datenbank für unsere Temperaturwerte zu erstellen. Mit dem Befehl `create database digidb` erstellen wir unsere Datenbank mit dem Namen *digidb* und mit dem Befehl `exit` können wir den Monitor wieder verlassen, da wir die weiteren Befehle auf eine andere Art eingeben werden.

Anmerkung:

Bei *MySQL* gibt es einen Administrator, der in der Regel auch mit *root* bezeichnet wird. Bei der Grundinstallation hat dieser kein Passwort. Das werden wir gleich ändern. Mit dem Befehl: `mysqladmin -u root -p password geheim` ändern wir das Passwort für den User *root* auf *geheim*.

Um die Änderungen wirksam zu machen, muss die Verwaltungstabelle neu eingelesen werden, dies erledigt der Befehl: `mysqladmin -u root -p flush-privileges`. Ab jetzt ist für jeden Zugriff auf die Datenbank als User *root* ein Passwort erforderlich.

Die Eingabe der Befehle über den Monitor ist sehr umständlich, das Programm *MySQL* bietet aber auch eine andere Möglichkeit, Befehle einzulesen.

Hierzu wird eine Textdatei mit den SQL Befehlen einfach mit < an den Befehl *MySQL* weitergeleitet.

Für ein Beispiel erstellen wir uns einfach eine Textdatei mit der wir die erste Tabelle für den Sensor 0 erstellen.

In der Datei *sensor0.sql* schreiben wir nun die Befehle zum Erstellen der Tabelle, diese könnte so aussehen.

```
CREATE TABLE sensor0 (
  id int(11) NOT NULL auto_increment,
  monat char(3) NOT NULL default '',
  tag char(2) NOT NULL default '',
  dbtime timestamp(14) NOT NULL,
```

```
  zeit time NOT NULL default '00:00:00',
  messung decimal(4,2) NOT NULL default '0.00',
  PRIMARY KEY  (id)
) TYPE=MyISAM;
```

Eingelesen wird diese dann mit:
`mysql -u digitemp -p digitemp < sensor0.sql`

Da wir 2 Sensoren verwenden, müssen wir die Datei nur noch kopieren und die Zeile `CREATE TABLE sensor0` in `CREATE TABLE sensor1` ändern.

Spätestens jetzt wird klar, dass die Verarbeitung der SQL Befehle über einer Datei, Vorteile bringt.

Kontrolle:

Um die neu erstellten Tabellen anzuzeigen, verwenden wir den Befehl: `echo 'show tables' | mysql -u root -p digidb` Ja, es geht auch anders rum.

Wenn wir alles richtig gemacht haben, werden wir mit der Ausgabe:

```
  Enter password:
  Tables_in_digidb
  sensor0
  sensor1
```

belohnt.

Datenbank mit Daten füllen

Das Füllen der Datenbank übernimmt eine kleines Perl Programm. Hier kommt auch unser erstes Perlmodul (DBI) zum Einsatz, das uns Methoden für den Datenbankzugriff bereitstellt.

Anmerkung:

Perl Module für alle Bereiche findest du im 'Comprehensive Perl Archive Network (CPAN)' unter der Adresse http://www.cpan.org/. Die Beschreibung der Installation spare ich mir und verweise auf die Seiten:
http://www.pro-linux.de/news/2002/0070.html
oder
http://www.linux-magazin.de/Artikel/ausgabe/1997/10/CPAN/cpan.html.

```
#!/usr/bin/perl -w
#
# Digitemp Logdatei aufbereiten und in Datenbank sichern
# sbs 2003-08-09
#
use DBI;
use strict;

# Datenbank initialisieren
my $datasource = "dbi:mysql:database=digidb";
my $user = "root";
my $pass = "geheim";

my $db = DBI->connect($datasource, $user, $pass)
   or die "Verbindung zur Datenbank nicht möglich: " . $DBI::errstr;

# Digitemp filtern
```

```
while(<STDIN>) {
  chomp;
  # Ausgabe Programname überspringen
  next if (m/Digi.*/);
  # Ausgabe Leerzeile überspringen
  next if (m/^$/);
  # alles nach Fahrenheit überspringen
  m/(.*).F.*/;
  my $tempzeile = $1;

  # Tempzeile aufteilen und in  Variablen speichern
  my ($monat, $tag, $zeit, $sensor_txt, $sensor_nr, $grad_txt, $grad_wert)
  = split(/ /,$tempzeile);

  # Datenbank füllen
  $db->do( "insert into sensor$sensor_nr (monat, tag, zeit, messung)
  values ('$monat', '$tag', '$zeit','$grad_wert')")
    or die "do nicht möglich: " . $db->errstr();

}# END- Digitemp filter

# Datenbank schliessen
$db->disconnect;
```

Kurze Erklärung zum Programm:

Das Programm macht eigentlich nicht viel, es öffnet nur die Datenbank, liest die Ausgabe, die es von *digitemp* bekommen hat, filtert alles raus, was wir nicht brauchen und schreibt dann die relevanten Werte in die jeweilige Datenbanktabelle.

Die laufende Erfassung der Werte erledigt uns der altbewährte cronjob:

```
  0-59/15 * * * * root /root/bin/digitemp -a | /root/bin/digipipe.pl
```

Das war es auch schon mit der Erfassung, jetzt geht es an das Web−Interface.

Perl und CGI

Auch hier bietet uns Perl die richtige Umgebung, um diese Aufgabe zu bewältigen.

Als erstes brauchen wir das Verzeichnis, in dem Apache seine CGI Programme verarbeitet. Finden kannst du diese in der Konfigurationsdatei von Apache. Suche nach einem Eintrag ähnlich diesem `<Directory /usr/lib/cgi-bin>`.

Bevor wir mit der grafischen Ausgabe beginnen, erstellen wir uns zuerst ein Programm, das uns die letzten gemessenen Werte ausgibt.

Am besten legst du diese in einem separaten Unterverzeichnis ab, außerdem musst du beachten, dass dein Programm ausführbar ist `chmod 755 programmname`.

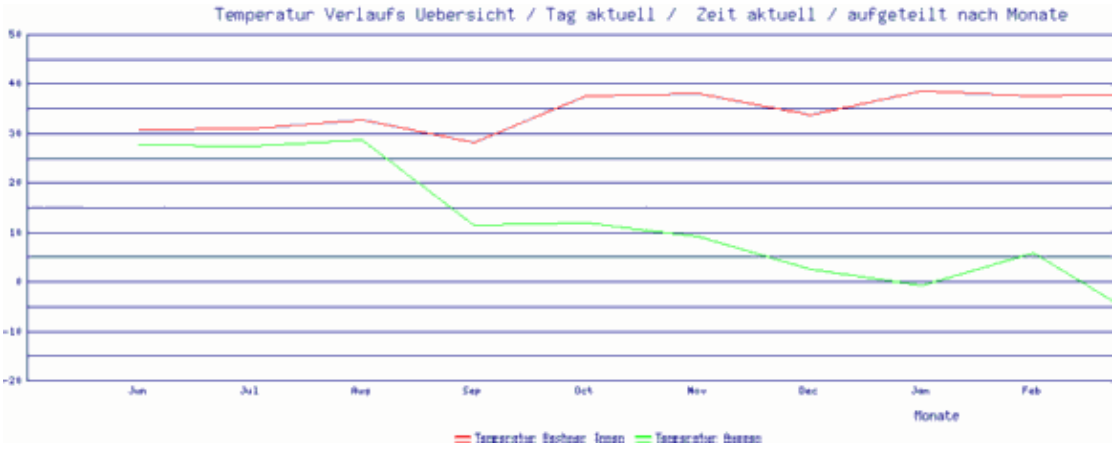
Die ganze Kunst der Ausgabe liegt in der jeweiligen SQL Abfrage. Wir müssen nur die Ausgabe auf den letzten Wert beschränken und dieses in ein Perl−CGI Programm stecken.

```
#!/usr/bin/perl

use DBI;
use strict;
```

```
# Datenbank initialisieren
my $datasource = "dbi:mysql:database=digidb";
my $user = "root";
my $pass = "geheim";

my $db = DBI->connect($datasource, $user, $pass)
   or die "Verbindung zur Datenbank nicht möglich: " . $DBI::errstr;

# Datenbank Arbeitsparameter
my $sql;
my $sth;

# Sensor Arbeitsparameter
my $temp;
my $zeit;

# HTML Ausgabe vorbereiten
print "Content-type: text/html\n\n";

# Werte der einzelnen Sensoren ausgeben
for (my $i=0; $i<2; $i++) {
  $sql = "select messung, zeit from sensor$i order by id desc limit 1;";

  $sth = $db->prepare($sql)
    or die "prepare nicht möglich";
  $sth->execute()
    or die "execute nicht möglich";
  ($temp, $zeit) = $sth->fetchrow_array();
  $sth->finish();

  print "<p>Temperatur Sensor$i: <b>[$temp]</b> $zeit</p>";

}

# Datenbank schliessen
$db->disconnect;
```

Das Beispiel ist jetzt nicht das eleganteste, aber es soll ja nur zeigen, wie einfach diese Aufgabe mit Perl zu bewältigen ist.

Grafische Ausgabe



Wagen wir uns nun an die grafische Ausgabe. Das Programm erstellt Liniengrafiken, wer andere Grafiken benötigt sollte sich die anderen GD Module ansehen.

Außerdem wird im Programm das CGI Modul verwendet, mit dem HTML Ausgaben unter Perl erstellt werden können. Hier verweise ich auch auf die vielen Beschreibungen im Internet.

Aber jetzt wieder zum Programm. Es besteht aus einem Hauptteil und 2 Unterprogrammen. Das eine Unterprogramm ist für die SQL Abfrage, das zweite für die grafische Darstellung zuständig.

Im Hauptteil werden nur mehr 3 Abfragen erstellt und die Daten an die Unterprogramme weitergegeben.

1. Erstellen der Einteilung der X−Achse
2. Daten des 1. Sensors (sensor0)
3. Daten des 2. Sensors (sensor1)

Es brauchen also nur die Abfragen geändert zu werden, um unterschiedliche grafische Auswertungen zu erstellen.

SQL Abfragen

Zum Schluss möchte ich euch noch ein paar SQL Abfragen zeigen, diese sind ja Dreh− und Angelpunkt der ganzen Beschreibung.

Die letzten 5 Messungen

```
select tag, monat, zeit,
  DATE_FORMAT(dbtime,'%Y-%c-%d %H:%i:%s') as dbtime, messung
    from sensor0
    order by id desc
      limit 5;
```

Der kälteste Tag im Jahr

```
select tag, monat, zeit,
  DATE_FORMAT(dbtime,'%Y-%c-%d %H:%i:%s') as dbtime, messung
    from sensor1
    where YEAR(dbtime) = YEAR(NOW())
      order by messung asc
      limit 1
```

Der wärmste Tag im Jahr

```
select tag, monat, zeit,
  DATE_FORMAT(dbtime,'%Y-%c-%d %H:%i:%s') as dbtime, messung
    from sensor1
    where YEAR(dbtime) = YEAR(NOW())
      order by messung desc
      limit 1
```

arithmetisches Mittel (Durchschnittswert) des Tages berechnen

```
select tag, monat, YEAR(dbtime) as Jahr,
  sum(messung)/count(*) as Durchschnitt
     from sensor1
    where YEAR(dbtime) = YEAR(NOW())
    and DAYOFMONTH(dbtime)= DAYOFMONTH(NOW())
    and MONTHNAME(dbtime) = MONTHNAME(NOW())
      group by DAYOFMONTH(dbtime)
```

Schlusswort

Ich selbst bin immer wieder fasziniert, wie einfach es ist, mit Perl Programme zu schreiben. Eigentlich kann man gar nicht sagen schreiben, man muss sich die Teile nur zusammenkopieren, irgendwie gibt es alles schon in irgendeiner Form.

Ich hoffe, ich konnte euch einen kleinen Einblick in die Thematik Perl, CGI, MySQL geben.

Download

- CGI Programm „Aktuelle Werte ausgeben``
- CGI Programm „Grafische Darstellung bezogen auf den aktuellen Tag und der abgefragten Stunde``

Links / Referenzen

- Homepage CPAN
- Artikel Temperaturüberwachung mit Linux
- Beschreibung Installation CPAN Module// oder
- Referenz (de) mysql
- Homepage Perl