Surveillance de la température avec Linux (2)

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

L´auteur:

Electricien qualifié, je me suis retrouvé, en 1990, devant une station de travail de DAO pour développer une station de commutateur et de commande. Evidemment, j'ai été infecté par un « virus » inconnu alors et c'était parfait.

Traduit en Français par:
Jean–Etienne Poirrier
(homepage)

Résumé:

Dans l'édition de novembre 2003, Surveillance de température avec Linux, je détaillais un circuit qui nous permet de collecter les données de température avec Linux. Pour évaluer ces données de température, nous devons les stocker dans une base de données.

Pour obtenir le maximum de bénéfices de cet article, nous allons afficher graphiquement les données à travers une interface web.

Pré–requis

Quelques applications courantes devraient déjà être installées sur votre ordinateur :

- Perl
- Apache
- MySQL
- et quelques modules Perl qui rendront la génération de programmes plus facile (ce sujet est développé plus tard).

Comme nous l'indiquons, cet article est destiné aux utilisateurs un tant soit peu avancés de Linux. Pour ceux qui ne sont pas encore à ce stade, cela sera une bonne introduction ;–)

Mise en place de la base de données

Dans MySQL, le programme *mysql* fournit l'interface à la base de données. Avec la commande `mysql -u root -p mysql`, nous nous connectons au moniteur de MySQL.

Avec l'option `-u`, l'utilisateur peut être saisi. L'option `-u` va demander un mot de passe et, finalement, la base de données à utiliser doit être saisie. Dans notre cas, nous sélectionnons le Gestionnaire de base de données de MySQL.

Vous allez recevoir alors l'invite `mysql >` pour y entrer des commandes SQL. Tout d'abord, nous devons trouver quel type de tables se trouve dans la base de données. La commande `show tables;` fait cela.

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

Maintenant, nous devons créer la base de données pour nos données de température. Avec la commande `create database digidb`, nous générons notre base de données nommée *digidb* et avec la commande `exit` , nous pouvons quitter le gestionnaire car les commandes additionnelles vont être entrées d'une autre manière.

Note:

MySQL possède un administrateur qui, en général, est aussi nommé *root*. L'installation par défaut ne requiert pas de mot de passe. Avec la commande `mysqladmin -u root -p mot de passe secret`, nous changeons le mot de passe pour l'utilisateur *root* en *secret*.

Pour rendre active cette modification, la table de l'administrateur doit être lue de nouveau ; nous faisons cela avec la commande `mysqladmin -u root -p flush-privileges`. A partir de maintenant, l'utilisateur *root* doit donner un mot de passe pour chaque accès à la base de données.

Entrer des commandes via la ligne de commande est très compliqué. Cependant, *MySQL* offre une autre possibilité de le réaliser.

Pour ce faire, nous entrons un fichier texte avec des commandes SQL en ajoutant un « < » à la commande *mysql*.

Pour montrer ceci, nous écrivons un fichier texte qui générera la première table pour le capteur 0.

Dans ce fichier, *sensor0.sql*, nous écrivons les commandes pour créer la table ; cela pourrait ressembler à ceci.

```
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;
```

Cela sera entré avec :
`mysql -u digitemp -p digitemp < sensor0.sql`

Puisque nous utilisons 2 capteurs, nous ne devons que copier le fichier et modifier la ligne `CREATE TABLE sensor0` en `CREATE TABLE sensor1`.

A ce niveau, il devrait être évident qu'entrer les commandes SQL à l'aide d'un fichier a de réels avantages.

Vérification :

Pour afficher les tables nouvellement créées, nous utilisons la commande : `echo 'show tables' | mysql -u root -p digidb` ; bien sûr, cela fonctionne également par d'autres méthodes.

Si nous avons réalisé tout correctement, nous verrons la sortie suivante :

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

Entrer des données dans notre base de données

Un petit programme Perl va effectuer le transfert des données dans la base de données. Pour cela, notre premier module Perl (DBI) va être utilisé : il va fournir les méthodes d'accès à la base de données.

Note:

Des modules Perl pour tout type d'applications peuvent être trouvés dans le « Comprehensive Perl Archive Network (CPAN, http://www.cpan.org/) ». Je passe la description de l'installation et vous renvoie vers :
http://www.pro-linux.de/news/2002/0070.html
ou
http://www.linux-magazin.de/Artikel/ausgabe/1997/10/CPAN/cpan.html
.

```
#!/usr/bin/perl -w
#
# Digitemp préparation du fichier de log et sauvegarde dans une base de données

# sbs 2003-08-09
#
use DBI;
use strict;

# Initialisation de la base de données

my $datasource = "dbi:mysql:database=digidb";
my $user = "root";
```

```
my $pass = "secret";

my $db = DBI->connect($datasource, $user, $pass)
   or  "Connection à la base de données impossible : " . $DBI::errstr;

# Filtration de Digitemp
while(<STDIN>) {
  chomp;
  # Passer le nom du programme (de l'output)
  next if (m/Digi.*/);
  # Passer la ligne blanche (dans l'output)
  next if (m/^$/);
  # Passer tout jusque Fahrenheit (dans l'output)
  m/(.*).F.*/;
  my $templine = $1;

  # Divise la ligne temp et sauvegarde dans des variables
  my ($monat, $tag, $zeit, $sensor_txt, $sensor_nr, $grad_txt, $grad_wert)
  = split(/ /,$tempzeile);

  # Remplir la base de données
  $db->do( "insert into sensor$sensor_nr (monat, tag, zeit, messung)
  values ('$monat', '$tag', '$zeit','$grad_wert')")
    or die "Pas possible :  " . $db->errstr();

}# END- Filtre Digitemp

# Ferme la base de données
$db->disconnect;
```

Une brève explication du programme :

Le programme ne fait pas réellement grand chose : il ouvre la base de données, lit la sortie qu'il reçoit de *digitemp*, filtre tout ce dont nous n'avons pas besoin et écrit les données intéressantes dans la table correcte de la base de données.

La collecte continue des données est réalisée avec le travail éprouvé de cron :

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

C'est tout pour la collecte des données ; maintenant, au tour de l'interface.

Perl et CGI

Perl nous offre l'environnement approprié pour cette tâche.

Tout d'abord, nous devons connaître le répertoire où Apache exécute ses programmes CGI. Cela peut être trouvé dans les fichiers de configuration d'Apache. Cherchez une ligne comme celle–ci : `<Directory /usr/lib/cgi-bin>`.

Avant de commencer avec la sortie graphique, nous allons d'abord écrire un programme qui nous fournit les dernières données mesurées.

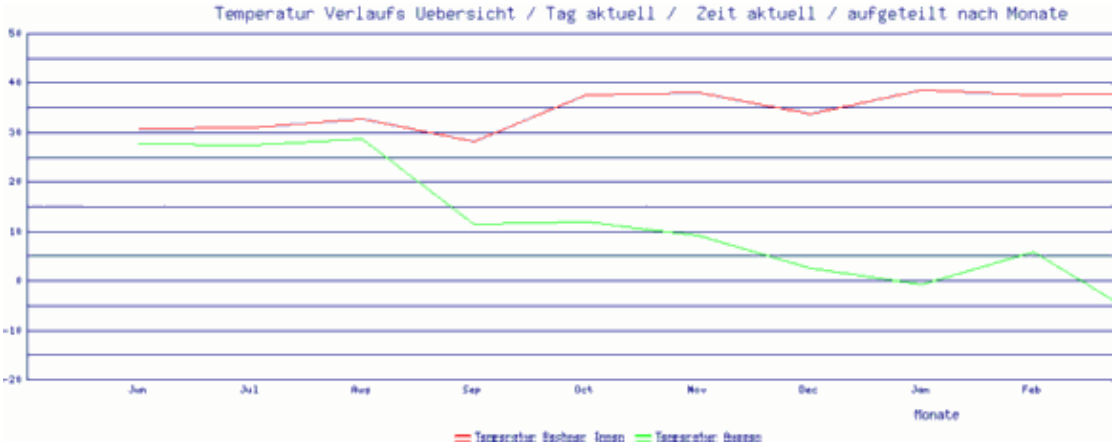
Il serait intéressant de stocker ces données dans un sous–répertoire séparé ; vous devez aussi rendre votre programme exécutable : `chmod 755 nomduprogramme`.

Nous devons limiter la sortie aux dernières données et les entrer dans un programme Perl–CGI. Cela sera fait avec une requête SQL.

```
#!/usr/bin/perl

use DBI;
use strict;

# Initialisation de la base de donnée
my $datasource = "dbi:mysql:database=digidb";
my $user = "root";
my $pass = "secret";

my $db = DBI->connect($datasource, $user, $pass)
   or  "Connection à la base de donnée impossible : " . $DBI::errstr;

# Paramètres de travail de la base de données
my $sql;
my $sth;

# Paramètre de travail des senseurs
my $temp;
my $zeit;

# Préparation de la sortie HTML
print "Content-type: text/html\n\n";

# Sortie des mesures des senseurs individuels
  $sql = "select messung, zeit from sensor$i order by id desc limit 1;";

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

  print "<p>Temp Senseur$i: <b>[$temp]</b> $zeit</p>";

}

# Fermer la base de données
$db->disconnect;
```

Cet exemple n'est pas le plus élégant ; il est seulement pour démontrer la manière dont ce travail peut être accompli simplement avec Perl.

Sortie Graphique



Maintenant, attaquons−nous à la sortie graphique. Le programme (à télécharger à la fin de cet article) génère la courbe graphique ; pour en savoir plus sur les graphiques, regardez les autres modules GD.

De plus, le programme utilisent le module CGI qui permet une sortie HTML avec Perl. Je me réfère ici aux nombreuses descriptions à ce sujet sur Internet.

Retour au programme... Il comprend une partie principale et deux sous−programmes. Un sous−programme est responsable de la requête SQL, le second est responsable des graphiques.

Seulement trois requêtes sont effectuées par la partie principale et les données sont passées aux sous−programmes.

1. Générer l'échelle de l'axe des X
2. Données du premier Capteur (sensor0)
3. Données du second Capteur (sensor1)

Seules les requêtes doivent être modifiées pour générer différentes sorties graphiques.

Requêtes SQL

Finalement, je souhaite vous montrer quelques requêtes SQL car elles sont le sujet principal de cet exemple.

Les cinq dernières mesures

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

Le jour le plus froid de l'année

```
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
```

Le jour le plus chaud de l'année

```
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
```

Calculer la moyenne arithmétique du jour

```
select day, month, 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)
```

Conclusion

Je suis toujours surpris de la simplicité avec laquelle des programmes peuvent être écrits en Perl. En fait, ils ne sont pas réellement écrits mais copiés et les sections sont combinées. D'une certaine manière, tout cela existe déjà quelque part sous une forme ou une autre.

J'espère que j'ai pu vous fournir un petit aperçu sur les sujets de Perl, CGI et MySQL.

Téléchargement

- Programme CGI "prend les valeurs courantes"
  (Programme CGI "Sortie des Mesures actuelles")
- Programme CGI "représentation graphique, relative au jour / à l'heure courant(e)"
  (Programme CGI "Présentation Graphique pour le jour courant et le temps requis")

Liens / Références

- Page d'accueil de CPAN
- Article Surveillance de température avec Linux
- Description de l'installation d'un module CPAN // ou
- Référence (allemande) de MySQL
- Page d'accueil de Perl

2005−01−22, generated by lfparser_pdf version 2.51