LinuxFocus article number 331
http://linuxfocus.org

door Diego Alberto Arias Prad
<dariapra(at)yahoo.es>

Over de auteur:

Ik ben een telecommunicatie ingenieur in Lugo, Spanje. Ik kan me niet precies herinneren wanneet ik ben begonnen met Linux te gebruiken, het was in 1995 of 1996. Daarvoor was ik een Microsoft Windows gebruiker en wist niet eens dat Linux bestond. De eerste keer dat ik Linux zag draaien op een computer, was op de universiteit. Het zag er erg interessant uit en al snel installeerde ik het op mijn computer thuis; Ik herinner me dat mijn eerste Linux distro Slackware was.

In al deze jaren heb ik veel plezier gehad aan het gebruik van andere Linux distros als ook enkele BSD "smaken", met het gebruik

Een introductie tot de TclMySQL bibliotheek

Kort:

In dit artikel bekijken we de installatie en het gebruik van MySQLTcl, een Tcl library (bibliotheek) die het mogelijk maakt om SQL queries (select, insert, delete...) op een MySQL database server uit te voeren vanuit Tcl scripts. De versies van Tcl, MySQL server en de MySQLTcl die besproken worden in dit artikel zijn respectievelijk 8.4.2, 4.0.15 en 2.40.

Tcl staat voor Tool Command Language en is uitgevonden door John Ousterhout [1]. Tcl is eigenlijk twee dingen tegelijk: een scripttaal en een interpreter. Tcl is een gestructueerde programmeer taal welke drie basis data structuren gebruikt: strings, lijsten en arrays. Features van Tcl omvatten reguliere expressies [2], third party Tcl uitbreidingen en Tk, een toolkit voor het schrijven van grafische applicaties in Tcl.

MySQL is een erg populaire database server in de open source community en heeft volgens mij geen presentatie meer nodig.

MySQLTcl is een Tcl library die het mogelijk maakt om vanuit een Tcl script queries op een MySQL database server uit te voeren. Op het moment zijn de auteurs en ontwikkelaars van deze Tcl library Paolo

"smaken", met het gebruik van programmeertalen als Java of Tcl, met database, web en applicatie servers... Linux betekende niet alleen plezier voor mij, ik had ook de kans met Linux te werken toen ik werkte bij Telefónica I+D.

Brutti (Paolo.Bruti at tlsoft.it), Tobias Ritzau (tobri at ida.liu.se) en Artur Trzewick (mail at xdobry.de).

Vertaald naar het Nederlands door:
Guus Snijders
<ghs(at)linuxfocus.org>

MySQLTcl library installatie

Als je Linux distro of je *BSD besturingssysteem ondersteuning biedt voor pakketten (zoals RPM of DEB bijvoorbeeld) of ports (Crux Linux of FreeBSD, bijvoorbeeld), kun je dit pakket of ports systeem gebruiken om de MySQLTcl library te installeren en deze sectie overslaan.

Als dit niet het geval is, of als je voorkeur geeft aan "met de hand" installeren, kun je de stappen die ik heb gevolgd, gebruiken. Deze regels moet je zien als een gids en niet als een stap-voor-stap installatie handleiding. In een Linux Mandrake *distro* (versie 9.2), vanuit bash:

```
$ ./configure --with-mysql-lib=/usr/lib
$ make
$ make install
```

Als er iets misgaat tijdens de "configure" stap, zou de fout informatie aanwijzingen moeten leveren om het probleem op te lossen. Meestal is het probleem dat het *configure* script bepaalde directories of bestanden niet kan vinden. In dergelijke gevallen kun je met het script "spelen" door deze parameters mee te geven waar de missende bestanden of directories staan. Laten we eens kijken naar een ander voorbeeld. Toen ik FreeBSD 5.0 gebruikte, installeerde ik MySQLTcl met deze opties:

```
$ export CPP=/usr/bin/cpp
$ ./configure --with-tcl=/usr/lib/local/tcl8.3
  --with-tclinclude=/usr/local/include/tcl8.3
  --with-mysql-include=/usr/local/include/mysql
  --with-mysql-lib=/usr/local/lib/mysql
$ make
$ make install
```

Zoals je kon zien, was in dit tweede voorbeeld de versie van Tcl 8.3; de versie van de MySQLTcl library was trouwens 2.15 en de MySQL database server versie was 3.23.54.

Tcl Basics

In deze sectie introduceren we een paar Tcl basis technieken voor die lezers die wel in dit artikel zijn geïnteresseerd maar geen ervaring hebben met Tcl. Als je al een Tcl programmeur bent, kun je deze sectie overslaan.

Je kunt de voorbeelden in deze (en volgende) sectie reproduceren in de Tcl shell (tclsh).

Variabelen. Commando en variabele substitutie.

Tcl variabelen maak je met het commando *set*. Laten we een paar voorbeelden bekijken:

```
% set address {Edison Avenue, 83}
Edison Avenue, 83
% set zip_code 48631
48631
```

In deze twee voorbeelden hebben we twee variabelen gecreëerd, genaamd adress en zip_code (NL: postcode). De waarden van deze variabelen zijn, respectievelijk *Edison Avenue,* 83 en *48361*; beide waarden zijn strings. Merk op dat voor het creëren van het variabele adres accolades zijn gebruikt, omdat de string witruimtes (spaties) bevat. De waarde van een variabele kan achterhaald worden met het *set* commando:

```
% set address
Edison Avenue, 83
% set zip_code
48631
```

Stel dat we de waarde van een variabele op het scherm willen afdrukken. Dit kun je doen met het commando *puts*:

```
% puts stdout [set address]
Edison Avenue, 83
```

De parameter *stdout* wordt meegegeven aan het *puts* commando. Deze parameter verteld het commando om af te drukken op de standaard uitvoer, in ons geval het scherm. De tweede parameter die aan het *puts* wordt meegeven is *[set address]*. De vierkante haken in de tweede parameter vertellen de Tcl interpreter dat de data binnen de haken een ander Tcl commando is, welke door de Tcl interpreter moet worden uitgevoerd voor het *puts* commando; dit wordt commando substitutie genoemd. Hetzelfde kan op een andere manier worden gedaan:

```
% puts stdout $address
Edison Avenue, 83
```

Wat we in dit voorbeeld gedaan hebben, wordt variabele substitie genoemd: het karakter *$* voor de naam van de variabele zorgt ervoor dat de substitutie plaatsvindt.

In een eerder voorbeeld zagen we dat het gebruik van accolades rond woorden die gescheiden zijn door

spaties ervoor kan zorgen dat ze worden gegroepeerd in een string. Een andere vorm van groeperen gaat met behulp van dubbele aanhalingstekens ("). Deze twee vormen van groeperen werken echter niet hetzelfde. Een voorbeeld:

```
% puts stdout "de postcode is [set adress]"
de postcode is 48631
% puts stdout "de postcode is $address"
de postcode is 48631
% puts stdout {de postcode is [set address]}
de postcode is [set address]
% puts stdout {de postcode is $address}
de postcode is $address
```

In dit voorbeeld kun je zien dat het gebruik van accolades voor het groeperen van commando en variabele substitutie niet veroorzaakt; echter deze vinden wel plaats als er dubbele aanhalingstekens worden gebruikt voor het groeperen.

Variabelen worden verwijderd met het *unset* commando:

```
% unset address
% set address
can’t read "address": no such variable
% unset zip_code
% set zip_code
can’t read "zip_code": no such variable
```

Tcl Strings

De string data structuur is een van de drie basis Tcl data structuren. Een string is een set karakters. Een string kan gecreëerd worden met het *set* commando.

```
% set achternaam Westmoreland
Westmoreland
% set nummer 46.625
46.625
```

De *achternaam* en *nummer* variabelen zijn strings. Strings kunnen gemanipuleerd worden met het *string* commando. De syntax van het *string* commando is *string operatie stringwaarde anderargument*. De volgende voorbeelden laten zien hoe dit commando gebruikt kan worden:

```
% string length $achternaam
12
% set achternaam [string range $achternaam 0 3]
West
% puts stdout $achternaam
West
% string tolower $achternaam
west
```

In tegenstelling tot Java of Pascal is Tcl geen sterk-type programmeer taal. Het volgende voorbeeld laat dit zien:

```
% set nummer [expr $nummer + 24.5]
```

```
70.125
% string range $nummer 2 5
.125
```

Met het commando *expr* werd de waarde van de variabele *nummer* verhoogd met 24,5. Daarna werd de variabele *nummer* behandeld als een string en werden de laatste vier karakters weergegeven.

Uiteraard zijn er meer string operaties mogelijk dan voorgaande voorbeelden.

Tcl Lijsten

Tcl lijsten zijn een speciaal geval van strings waarin de lijst elementen zijn gescheiden door witruimtes en speciaal geïntepreteerd worden.

```
% set vrienden_lijst {Fanny John Lisa Jack Michael}
Fanny John Lisa Jack Michael
% set vrienden_lijst [string tolower $vrienden_lijst]
fanny john lisa jack michael
% set vrienden_lijst
fanny john lisa jack michael
```

Er zijn een aantal Tcl commando's voor het manipuleren van lijsten. Een paar voorbeelden:

```
% lindex 2 $vrienden_lijst
lisa
% lrange $vrienden_lijst 2 4
lisa jack michael
% set vrienden_lijst [lsort -ascii $vrienden_lijst]
fanny jack john lisa michael
% set vrienden_lijst
fanny jack john lisa michael
% set vrienden_lijst [linsert $vrienden_lijst 2 Peter]
fanny jack Peter john lisa michael
% string toupper $vrienden_lijst
FANNY JACK PETER JOHN LISA MICHAEL
```

Het laaste voorbeeld geeft aan dat strings en lijsten in feite dezelfde datastructuren zijn.

Tcl Arrays

Een array kun je beschouwen als een lijst met een index van string-waarden. Een array kan als volgt gecreëerd worden:

```
% set telefoon_nummers(fany) 629
629
% set telefoon_nummers(michael) 513
513
% set telefoon_nummers(john) 286
286
```

Waarden uit de array kun je achterhalen met het *set* commando en variabele substitutie; zoals te zien is in het vorige voorbeeld, is een string-waarde index gescheiden door haakjes.

```
% set $telefoon_nummers(michael)
513
```

Het commando *array* geeft informatie over een array variabele. Laten we een paar voorbeelden bekijken over wat dit commando kan doen:

```
% array size telefoon_nummers
3
% array names telefoon_nummers
fany john michael
```

Database connectie

Alvorens een query op een database uit te voeren vanuit een Tcl script, moet er eerst een verbinding met de database server worden opgezet. In deze sectie zullen we zien hoe we dat kunnen doen en hoe we eventuele fouten, die kunnen optreden bij het maken van de database connectie, kunnen afhandelen.

Een database connectie opzetten

Laten we eens een voorbeeld bekijken van een Tcl script die een verbinding opzet met een MySQL database server:

```
0:  #!/usr/bin/tclsh8.4
1:
2:  package require mysqltcl
3:  global mysqlstatus
4:
5:  set port {3306}
6:  set host {127.0.0.1}
7:  set user {john_smith}
8:  set password {jsmith_password}
9:
10: set mysql_handler [mysqlconnect -host $host
    -port $port -user $user -password $password]
11:
12: mysqlclose $mysql_handler
```

Merk op dat de nummers in de linker kolom en de bijbehorende dubbele punt geen deel zijn van het Tcl script; het enige doel is het labelen van de Tcl script regels. Merk ook op dat regel #0 mogelijk, afhankelijk van je Linux distributie, gewijzigd moet worden om een geldig pad naar de Tcl shell aan te geven.

Regel #0 verteld de shell dat dit bestand een Tcl script is en waar het de Tcl interpreter kan vinden.

Regel #2 verteld de Tcl interpreter om in de MySQLTcl library te zoeken voor het uitvoeren van de

commando’s in het script. Zo zien we in regel #10 bijvoorbeeld het commando *mysqlconnect*; als regel #2 niet was opgenomen in het script, zou de Tcl interpreter regel #10 uitvoeren en het commando *mysqlconnect* zou een "command not found" (NL: opdracht niet gevonden) foutmelding geven.

In de regels #5 en #6 worden de poort en de host ingesteld, waarop het Tcl script verbinding maakt. In dit script is het poort nummer 3306 (de standaard poort voor MySQL) en de host is dezelfde machine als waarop het Tcl script wordt uitgevoerd.

In de regels #7 en #8 worden de database gebruiker en het bijbehorende wachtwoord ingesteld.

Op regel #10 wordt de eigenlijke database connectie opgezet. De uitvoer van het commando *mysqlconnect* wordt opgeslagen in de variabele die we hier *mysql_handler* hebben genoemd. Deze variabele zal de handler zijn voor de database connectie. De handler wordt gebruikt voor de queries op de database en ook voor het sluiten van de database connectie, zoals te zien op regel #12.

Fouten afhandelen

In de vorige subsectie, werd regel #3 van het script niet uitgelegd. Dat zullen we nu doen.

MySQLTcl library commando’s kunnen fouten opleveren. Als een fout niet wordt afgevangen, zal het script afbreken en het effect kan (wel of niet) interessant zijn voor ons. Laten we het script dat in de vorige subsectie werd getoond als volgt aanpassen:

```
10: if [catch {mysqlconnect -host $host -port $port
    -user $user -password $password} mysql_handler] {
11:     puts stderr {error, the database connection could not be established}
12:     exit
13: } else {
14:     mysqlclose mysql_handler
15: }
```

Als het commando *set mysql_handler [mysqlconnect -host $host...* een fout geeft, zal de foutmelding worden opgevangen door *catch*. Het commando *catch* geeft een 1 terug als het uitgevoerde commando binnen de accolades ({}) een fout opleverd of 0 als er geen fout optrad. De variabele *mysql_handler* slaat de uitvoer van het uitgevoerde commando, binnen de accolades, op.

Als er een fout optreed, wordt er een bericht afgedrukt op de standard error uitvoer (stderr), in ons geval het scherm. Er zijn een aantal oorzaken die een foutmeldinging kunnen opleveren bij het proberen een verbinding op te zetten met de database: fout wachtwoord, verkeerde host of poort... In dit geval is het handig om meer informatie te krijgen dan een simpele "er was een fout".

Op regel #3 werd de variabele *mysqlstatus* globaal gedefinieerd. Een globale variabele is er een die bereikbaar is vanuit iedere deel van een Tcl script; dit is gerelateerd aan het Tcl variabele bereik, een onderwerp waar we in dit artikel niet verder op ingaan. De library MySQLTcl creëert en onderhoud een globale array, genaamd *mysqlstatus*. Deze array heeft de volgende elementen:

element	betekenis

code	Als er geen fout was, dan is *mysqlstatus(code)* gelijk aan nul; anders, is *mysqlstatus(code)* gelijk aan de foutcode van de MySQL server. Als er een fout optrad die niet van de MySQL server kwam, word *mysqlstatus(code)* gezet op -1. De waarde van *mysqlstatus(code)* wordt iedere keer na het uitvoeren van een MySQLTcl library commando geüpdate.
command	Het laatste MySQLTcl library commando dat een fout gaf, is opgeslagen in *mysqlstatus(command)* De waarde van *mysqlstatus(command)* wordt na iedere niet-succesvolle uitvoering van een MySQLTcl commando geüpdate; *mysqlstutus(commmand)* wordt dus **niet** geüpdate na een succesvolle uitvoering van een MySQLTcl commmando.
message	De waarde van *mysqltcl(message* wordt na iedere niet-succesvolle uitvoering van een MySQLTcl commando geüpdate, met een string die het foutbericht bevat. Net als *mysqlstatus(command)*, wordt mysqlstatus(message) **niet** geüpdate na een succesvolle uitvoering van een MySQLTcl commando.

Er is een ander element in de globale arry *mysqlstatus* die niet gerelateerd is aan het behandelen van fouten:

element	betekenis
nullvalue	String die gebruikt wordt om de null waarde te representeren bij het weergeven van SQL query resultaten. Standaard wordt er een lege string gebruikt; *musqlstatus(nullvalue)* kan op iedere string waarde worden gezet.

Dus, door de globale array *mysqlstatus* te gebruiken, kan het vorige stuk code op deze manier worden herschreven om meer informatie te geven als er een fout optreedt tijdens het opzetten van een database connectie:

```
10: catch {mysqlconnect -host $host -port $port
    -user $user -password $password} mysql_handler
11: if {$mysqltatus(code) != 0} {
12:     puts stderr $mysqlstatus(message)
13: } else {
14:     mysqlclose mysql_handler
15: }
```

Uiteraard kan de globale array *mysqlstatus* worden gebruikt voor het afhandelen van meer fouten dan allen die die kunnen optreden tijdens het opzetten van de database connectie.

Basis MySQLTcl library commando’s

In deze sectie bekijken we de meest basis MySQLTcl library commando’s en zullen we aan de hand van voorbeelden zien hoe deze te gebruiken zijn. Voor een complete referentie, zie de MySQLTcl library man pagina.
De commando’s die in deze sectie beschreven worden, zijn weergegeven in de volgende tabel.

Parameters zijn onderstreept. Als een parameter tussen twee ? karakters staat, betekend dat dat deze optioneel is; het karakter | betekend "of".

commando	korte beschrijving
mysqlconnect ?optionele waarde ...?	verbindt met een database; een connectie handler die gebruikt kan worden door andere mysqltcl commando's wordt geretourneerd.
mysqluse handle dbnaam	associeert een MySQL handler met de opgegeven database
mysqsel handle sql_statement ?-list | -flatlist?	stuurt een geselecteerd SQL statement naar de database
mysqlexec handle sql_statement	stuurt een niet-select SQL statement naar de database
mysqlclose handle	sluit een database connectie

mysqlconnect

Dit commando is reeds ter sprake gekomen in de sectie "Database connectie". Dit commando accepteert een extra parameter, die hier nog niet is besproken: -db. Met de parameter -db, wordt altijd de hiermee aangegeven database gebruikt in komende SQL statements. Laten we eens kijken naar een voorbeeld:

```
% set mysql_handler [mysqlconnect -h 127.0.0.1 -p 3306 \\
  -user john_smith -password jsmith_password -db jsmith_database]
```

De "doel" database van MySQLTcl commando's die de *mysql_handler* database handler gebruiken, zullen gebruik maken van degene die *jsmith_database* genoemd wordt.

(Merk op dat de karakters \\ geen onderdeel zijn van het commando; deze geven aan dat het commando verder gaat op de volgende regel.)

mysqluse

Dit commando staat je toe om de database die werd geassocieerd werd met de MySQL handler te veranderen, door de nieuwe database als eerste parameter mee te geven.

mysqlsel

Dit commando stuurt een SQL select statement naar de database die geassocieerd is met de MySQL handler. Als de parameter sql_statement geen SQL select statement is, zal er een fout optreden.

Er is een derde optionele parameter, deze kan list zijn of flat_list. Laten we eens in een voorbeeld bekijken hoe deze parameter de uitvoer van dit commando beïnvloed. Stel dat we in de database die genoemd is in de MySQL handler een tabel hebben met de naam *people*, zoals hieronder weergegeven:

id	first_name	last_ name	phone
26	Karl	Bauer	8245
47	James	Brooks	1093
61	Roberto	Castro Portela	6507

We gebruiken het *mysqlsel* commando om een SQL select statement naar de database te sturen:

```
% mysqlsel $mysql_handler {select first_name, last_name from people order by id asc}
{Karl Bauer} {James Brooks} {Roberto {Castro Portela}}
% mysqlsel $mysql_handler {select first_name, last_name from people order by id asc}
Karl Bauer James Brooks Roberto {Castro Portela}
```

In het eerste voorbeeld (-*list parameter*), retourneerd het commando een lijst waarvan de elementen lijsten zijn. In het tweede voorbeeld (-*flatlist parameter*), retourneerd het commando een enkele lijst, waarin alle elementen samengevoegd zijn.

Wat gebeurd er als het *mysqlsel* commando geen derde parameter krijgt? In dit geval, zal de uitvoer van het *mysqlsel* commando het aantal regels weergeven dat de query opleverde:

```
% mysqlsel $mysql_handler {select first_name, last_name from people order by id asc}
3
```

mysqlexec

Het *mysqlexec* commando stuurt een non-select SQL statement naar de database die geassocieerd is met de MySQL handler. Als de parameter *sql_statement* een SQL select statement is, zal er geen fout optreden, maar zal er ook niets gebeuren.

We nemen het voorbeeld uit de vorige subsectie:

```
% mysqlexec $mysql_handler {delete from people where id=26}
1
% mysqlsel $mysql_handler {select first_name, last_name, phone from people order by
{James Brooks 1093} {Roberto {Castro Portela} 6507}
% mysqlexec $mysql_handler \\
  {insert into people (id, first_name, last_name, phone) values (58, "Carla", "di Be
1
% mysqlsel $mysql_handler {select first_name, last_name, phone from people order by
{James Brooks 1093} {Carla {di Bella} 8925} {Roberto {Castro Portela} 6507}
```

Natuurlijk kunnen ook ander SQL queries dan delete (verwijderen) of insert (invoegen) met dit commando naar de database worden gestuurd. Zo kun je bijvoorbeeld een regel updaten:

```
% mysqlexec $mysql_handler {update people set phone=3749 where first_name="James"}
1
% mysqlsel $mysql_handler {select first_name, last_name, phone from people order by
{James Brooks 3749} {Carla {di Bella} 8925} {Roberto {Castro Portela} 6507}
```

Het commando *mysqlexec* geeft het aantal regels die betrekking hebben op het SQL non-select statement

dat naar naar de database werd gestuurd.

mysqlclose

Zoals we eerder hebben gezien, sluit het commando *mysqlclose* een database connectie. De parameter voor dit commando is de MySQL handler van de database connectie die we willen sluiten.

Overige MySQLTcl library commando's

De MySQLTcl library kent meer commando's dan de 5 die we hebben besproken in deze sectie. Met deze commando's kun je informatie ophalen over de databases, strings formatteren zodat ze geschikt zijn voor queries, het maken van geneste queries... Een goede referentie is de eigen man pagina van MySQLTcl, welke onderdeel is van de installatie van de MySQLTcl library.

Referenties

[1] Een enigzins sceptische blik op John K. Ousterhout en Tcl:
http://www.softpanorama.org/People/Ousterhout/index.shtml

[2] een tutorial over Tcl Reguliere expressies:
http://www.mapfree.com/sbf/tcl/book/select/Html/7.html

TclTutor is een vrije, interactieve applicatie voor het leren van Tcl:
http://www.msen.com/~clif/TclTutor.html

MySQL documentatie in verscheiden formaten (HTML, PDF..):
http://www.mysql.com/documentation/index.html

Site onderhouden door het LinuxFocus editors team © Diego Alberto Arias Prad "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Vertaling info: en --> -- : Diego Alberto Arias Prad <dariapra(at)yahoo.es> en --> nl: Guus Snijders <ghs(at)linuxfocus.org>

2005-01-14, generated by lfparser_pdf version 2.51