LinuxFocus article number 304
http://linuxfocus.org

De MySQL C API

door Özcan Güngör
<ozcangungor(at)netscape.net>

Over de auteur:
Ik gebruik Linux sinds 1997. Vrijheid, flexibiliteit en opensource. Dat zijn de eigenschappen die mijn hart hebben veroverd.

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

Kort:

In dit artikel zullen we kijken hoe de C APIs (Application Programming Interfaces) van MySQL gebruikt kunnen worden. Om dit artikel te begrijpen, is enig begrip van de volgende nodig:

- Variabelen in C
- Functies in C
- Pointers in C

Introductie

De C APIs worden samen met MySQL gedistribueerd en zijn opgenomen in de *mysqlclient* bibliotheek. Deze worden gebruikt om een verbinding met een database op te zetten en een query uit te voeren. Voorbeelden kunnen gevonden worden in de *clients* directory van de MySQL bron code.

MySQL C Variabele Types

De volgende variabele types zijn gedefinieerd in de MySQL bibliotheek. We hebben deze variabelen nodig om de MySQL functies te kunnen gebruiken. Variabelen worden later in detail uitgelegd maar de details zijn eigenlijk niet erg belangrijk voor de te schrijven code.

MYSQL
De volgende structuur is een communicatie handler die wordt gebruikt om te verbinden met een database.

```
typedef struct st_mysql {
  NET           net;            /* Communication parameters */
  gptr          connector_fd;   /* ConnectorFd for SSL */
  char          *host,*user,*passwd,*unix_socket,
                *server_version,*host_info,*info,*db;
  unsigned int  port,client_flag,server_capabilities;
  unsigned int  protocol_version;
  unsigned int  field_count;
  unsigned int  server_status;
  unsigned long thread_id;      /* Id for connection in server */
  my_ulonglong affected_rows;
  my_ulonglong insert_id;       /* id if insert on table with NEXTNR */
  my_ulonglong extra_info;              /* Used by mysqlshow */
  unsigned long packet_length;
  enum mysql_status status;
  MYSQL_FIELD   *fields;
  MEM_ROOT      field_alloc;
  my_bool       free_me;        /* If free in mysql_close */
  my_bool       reconnect;      /* set to 1 if automatic reconnect */
  struct st_mysql_options options;
  char          scramble_buff[9];
  struct charset_info_st *charset;
  unsigned int  server_language;
} MYSQL;
```

MYSQL_RES

Deze structuur representeerd de resultaten van een query, welke rijen opleverd. De geretourneerde data wordt een result-set (set resultaten) genoemd.

```
typedef struct st_mysql_res {
  my_ulonglong row_count;
  unsigned int  field_count, current_field;
  MYSQL_FIELD   *fields;
  MYSQL_DATA    *data;
  MYSQL_ROWS    *data_cursor;
  MEM_ROOT      field_alloc;
  MYSQL_ROW     row;            /* If unbuffered read */
  MYSQL_ROW     current_row;    /* buffer to current row */
  unsigned long *lengths;       /* column lengths of current row */
  MYSQL         *handle;        /* for unbuffered reads */
  my_bool       eof;            /* Used my mysql_fetch_row */
} MYSQL_RES;
```

MYSQL_ROW

Deze structuur is een type-veilige representatie van data in een rij. Je kunt deze niet gebruiken als een string die eindigt met een null karakter, omdat de data in deze string binair kan zijn en null karakters kan bevatten.

```
typedef struct st_mysql_field {
  char *name;                   /* Name of column */
  char *table;                  /* Table of column if column was a field */
  char *def;                    /* Default value (set by mysql_list_fields) */
```

```
  enum enum_field_types type;   /* Type of field. Se mysql_com.h for types */
  unsigned int length;          /* Width of column */
  unsigned int max_length;      /* Max width of selected set */
  unsigned int flags;           /* Div flags */
  unsigned int decimals;        /* Number of decimals in field */
} MYSQL_FIELD;
```

my_ulonglong
Het type gebruikt voor het aantal rijen en voor mysql_affected_rows(), mysql_num_rows(), and mysql_insert_id(). Dit type levert een bereik van 0 tot 1.84e19. Op sommige systemen zal een poging om een waarde van type my_ulonglong weer te geven, niet werken. Om een dergelijke waarde weer te geven, converteer deze naar unsigned long en gebruik het %lu prinf formaat. Voorbeeld:
printf(Number of rows: %lu\n", (unsigned long) mysql_num_rows(result));

typedef unsigned long my_ulonglong;

Verbinden met MySQL en een query maken

Ik ga er vanuit dat MySQL is geïnstalleerd en dat een gebruiker en een tabel zijn aangemaakt. In geval van probleem, zie de www.mysql.com website.

Zoals eerder gezegd, de MySQL bibliotheken zitten bij de mysqlclient bibliotheek. Voor het compileren van een MySQL programma, moet de *-lmysqlclient* compiler optie gebruikt worden. MySQL header bestanden zijn te vinden onder /usr/include/mysql (dit kan afhankelijk van je Linux distributie zijn). De header van je programma, zal er dan ongeveer zo uit moeten zien:

#include <mysql/mysql.h>

MySQL variabele types en functies zijn opgenomen in dit header bestand.

Vervolgens creëren we een variabele die wordt gebruikt om te verbinden met een database. Dit kan eenvoudig gedaan worden met:

MYSQL *mysql;

Alvorens te verbinden met een database, moeten we de volgende functie aanroepen om de mysql variabele te initiëren:

mysql_init(MYSQL *mysql)

Then the

```
MYSQL * STDCALL mysql_real_connect(MYSQL *mysql,
                                   const char *host,
                                   const char *user,
```

```
                                const char *passwd,
                                const char *db,
                                unsigned int port,
                                const char *unix_socket,
                                unsigned int clientflag);
```

De functie wordt aangeroepen om de verbinding met de database op te zetten. host is hostnaam van de MySQL server, user is als wie we willen verbinden, passwd is het wachtwoord en db de database waarmee we willen verbinden. port geeft het TCP/IP poortnummer aan van de MySQL server. Unix_socket is het connectietype en client-flag is de flag die zorgt dat MySQL zich op een ODBC-achtige manier gedraagd. In dit artikel is deze op 0 gezet. Deze functie retourneerd 0 als de verbinding is opgezet.

Nu kunnen we verbinden met een database en een query gebruiken:

```
char *query;
```

Met behulp van deze string kunnen we willekeurige SQL zinnen construeren en een query maken. De functie om de query uit te voeren is:

```
int STDCALL mysql_real_query(MYSQL *mysql,
                             const char *q,
                             unsigned int length);
```

mysql is de variabele die we boven hebben gebruikt. q is de SQL query string. length is de lengte van deze string. Als de query zonder foutloos eindigt, retourneerd de functie 0.

Na het maken van een query, hebben we een variabele in MYSQL_RES nodig, om in staat te zijn gebruik te maken van de query resultaten. De volgende regel creëert deze variabele:

```
MYSQL_RES *res;
```

Dan

```
mysql_use_result(MYSQL *query)
```

De functie wordt gebruikt om resultaten te lezen.

Hoewel we eenvoudig queries kunnen maken, hebben we andere functies nodig om de resultaten te gebruiken. De eerst is:

```
MYSQL_ROW STDCALL mysql_fetch_row(MYSQL_RES *result);
```

Deze functie transformeert resultaten in rijen. Zoals je zult opmerken, retourneerd deze functie een MYSQL_ROW type variabele. De volgende regel creëert zo’n variabele:

```
MYSQL_ROW row;
```

Zoals hierboven uitgelegd is een variabele rij een array strings. Dit betekend dat row[0] de eerste kolom van de eerste rij is, row[1] is de tweede kolom van de eerste rij... Als we gebruik maken van mysql_fetch_row, krijgt de variabele row de data van de volgende rij uit het resultaat. Als we het einde van de resultaten bereiken, wordt er een negatieve waarde geretourneerd. Ten slotte moeten we de

verbinding sluiten:

```
mysql_close(MYSQL *mysql)
```

Wat Handige Functies

Laten we eens kijken hoe we het aantal velden in een tabel kunnen achterhalen. De volgende functie doet dit:

```
unsigned int STDCALL mysql_num_fields(MYSQL *mysql);
```

Deze functie retourneerd het aantal velden in de tabel.

Om het aantal rijen van het query resultaat te krijgen, gebruik:

my_ulonglong STDCALL mysql_num_rows(MYSQL_RES *res);

my_ulonglong STDCALL mysql_affected_rows(MYSQL *mysql);

Deze functie wordt gebruikt om het aantal rijen te achterhalen, die worden beïnvloed door INSERT, DELETE en UPDATE queries. Merk op dat de functie het type my_ulonglong gebruikt.

Ten slotte nog wat voorbeeld code:

```
#include <mysql/mysql.h>
#include <stdio.h>

void main(){
   MYSQL *mysql;
   MYSQL_RES *res;
   MYSQL_ROW row;
   char *query;
   int t,r;

   mysql_init(mysql);
   if (!mysql_real_connect(mysql,"localhost","mysql",
        "mysql","deneme",0,NULL,0))
   {
       printf( "Error connecting to database: %s\n",mysql_error(mysql));
   }
   else printf("Connected...\n");

   query="select * from Deneme";

   t=mysql_real_query(mysql,query,(unsigned int) strlen(query));
   if (t)
   {
      printf("Error making query: %s\n",
              mysql_error(mysql));
   }
   else printf("Query made...\n");
   res=mysql_use_result(mysql);
   for(r=0;r<=mysql_field_count(mysql);r++){
```

```
            row=mysql_fetch_row(res);
            if(row<0) break;
            for(t=0;t<mysql_num_fields(res);t++){
                    printf("%s ",row[t]);
            }
            printf("\n");
    }
    mysql_close(mysql);
}
```

Aanbevolen Literatuur

- De web site van MySQL: www.mysql.com
- Documenten die meekomen met de MySQL source. (waarschijnlijk onder de /usr/doc directory)

2005-01-14, generated by lfparser_pdf version 2.51