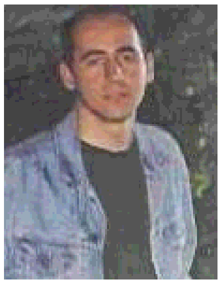
LinuxFocus article number 304
http://linuxfocus.org

The MySQL C API



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

About the author:
I use Linux since 1997. Freedom, flexibility and opensource. These are the properties I like.

Translated to English by:
Özcan Güngör
<ozcangungor(at)netscape.net>

Abstract:

In this article, we will learn how to use the C APIs (Application Programming Interfaces) that come with MySQL. In order to understand this article well you need knowledge about prerequisites:

- Variables in C
- Functions in C
- Pointers in C

Introduction

The C APIs are distributed with the MySQL source and included in the *mysqlclient* library. They are used to connect to a database and execute a query. There are some examples in the *clients* directory of the MySQL source code.

MySQL C Variable Types

The Following variable types are defined in the MySQL library. We need these variables in order to use the MySQL functions. Variables are explained in detail but the details are actually not so important to write code.

MYSQL
The following structure is communication handler that is used to connect to a 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
This structure represents the results of a query which returns rows. The data returned data is called result-set.

```
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
This structure is a type-safe representation of data in a row. You cannot use this as a string that ends with null character because data in this string can be binary and may include null character.

```
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
The type used for the number of rows and for mysql_affected_rows(), mysql_num_rows(), and mysql_insert_id(). This type provides a range of 0 to 1.84e19. On some systems, attempting to print a value of type my_ulonglong will not work. To print such a value, convert it to unsigned long and use the %lu printf format. Example:
printf(Number of rows: %lu\n", (unsigned long) mysql_num_rows(result));

typedef unsigned long my_ulonglong;

Connecting to MySQL and making a query

Now, I assume that MySQL is installed, a user and a table in a database is created. In case of any problem, please refer the www.mysql.com website.

As said before MySQL libraries are in the mysqlclient library. So while compiling a MySQL program, it is necessary to add the *-lmysqlclient* compiler option. MySQL header files are under /usr/include/mysql (this may depend on your Linux distribution). The header of you program has then to look like this:

#include <mysql/mysql.h>

MySQL variable types and functions are included in this header file.

Then, we need to create the variable that is used to connect a database. This is simply done with:

MYSQL *mysql;

Before connecting to a database, we must call the following function to initiate the mysql variable:

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);
```

function is called to connect to a database. host is the hostname of the MySQL server. user is the user we want to connect as. passwd is the password. db is the database we want to connect to. port is the TCP/IP port number of MySQL server. unix_socket is connection type. clientflag is the flag that makes MySQL run ODBC-like. In this article it is set to 0. This function returns 0 when connection is established.

Now we can connect to a database and make a query:

```
char *query;
```

Using this string we can construct any SQL sentences and make a query. The function that executes the query is:

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

mysql is the variable we have used above. q is the SQL query string. length is the lenth of this string. If the query suceeds without error, the function returns 0.

After making a query, we need a variable in MYSQL_RES in order to be able to use query results. The following line creates this variable:

```
MYSQL_RES *res;
```

Then

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

The function is used to read results.

Altough we can make queries easily, we need some other functions to use the results. The first one is:

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

This function tranforms results into rows. As you notice, the function returns a MYSQL_ROW type valiable. The following line creates such a variable:

```
MYSQL_ROW row;
```

As explained above variable row is an array of strings. This means that row[0] is the first column of the first row, row[1] is second column of the first row.... When we use mysql_fetch_row, then the variable row gets the data of the next row of the result. When we reach the end of the result, then the function returns a negative value. In the end we need to close the connection:

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

Some Useful Functions

Let’s see how to get the number of fields in a table. The following function does it:

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

This function returns the number of fields in the table.

To get the number of rows of query result use:

my_ulonglong STDCALL mysql_num_rows(MYSQL_RES *res);

my_ulonglong STDCALL mysql_affected_rows(MYSQL *mysql);

This function is used to know the number of rows that are affected by INSERT, DELETE, UPDATE queries. Note that the function returns the type my_ulonglong.

Finally some example 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);
}
```

Recommended Readings

- The web site of MySQL: www.mysql.com
- Documents coming along with the MySQL source. (Probably under the /usr/doc directory)
