A API em C do MySQL

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

About the author:
Eu uso o Linux desde 1997. Liberdade, flexibilidade e código aberto. Estas são as propriedades que gosto.

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

Abstract:

Neste Artigo, aprenderemos a usar as APIs em C (Interfaces de programação de aplicações) que vêm com o MySQL. No sentido de compreender este artigo precisa de ter conhecimento dos seguintes pré−requisitos:

- Variáveis em C
- Funções em C
- Ponteiros em C

Introdução

As APIs em C são distribuidas com o código do MySQL e estão incluídas na biblioteca do *mysqlclient*. São usadas para se ligar a uma base de dados e executar consultas. Existem alguns exemplos no directório *clients* do código fonte do MySQL.

Tipos de Variáveis do MySQL C

Os seguintes tipos de variáveis estão definidos na biblioteca do MySQL. Necessitamos destas variáveis no sentido de usar as funções do MySQL. As variáveis são explicadas em detalhe mas os detalhes não são agora importantes para escrever código.

MYSQL
A estrutura seguinte é o handler de comunicação que é usado para se ligar a uma base de dados.

```
typedef struct st_mysql {
NET           net;            /* Parâmetros de ligação */
  gptr          connector_fd;   /* Descriptor da ligação para 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 para a ligação no servidor */
  my_ulonglong affected_rows;
        my_ulonglong insert_id;       /* id da próxima inserção se for usado o NEXTNR */
  my_ulonglong extra_info;              /* Usado pelo mysqlshow */
  unsigned long packet_length;
  enum mysql_status status;
  MYSQL_FIELD   *fields;
  MEM_ROOT      field_alloc;
  my_bool       free_me;        /* Se está livre após mysql_close */
        my_bool       reconnect;      /* definido a 1 para uma re-ligação automática */
  struct st_mysql_options options;
  char          scramble_buff[9];
  struct charset_info_st *charset;
  unsigned int  server_language;
} MYSQL;
```

MYSQL_RES

Esta estrutura representa os resultados de uma consulta que retorna linhas. Os dados retornados chamam−se 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;            /* Se faz leitura recorrendo a buffers*/
        MYSQL_ROW     current_row;    /* buffer da actual linha */
        unsigned long *lengths;       /* tamanho das colunas da linha actual */
        MYSQL         *handle;        /* Para leituras que não recorrem a buffers */
  my_bool       eof;            /* Usado por mysql_fetch_row */
} MYSQL_RES;
```

MYSQL_ROW

Esta estrutura é uma representação de dados seguros numa linha. Não pode usar isto como uma string que termine com o caracter nulo porque os dados nesta string podem ser binários e podem incluir o caracter nulo.

```
typedef struct st_mysql_field {
  char *name;                   /* Nome da coluna */
        char *table;                  /* Tabela da coluna se a coluna for um campo */
  char *def;                    /* Valor por omissão (definido por mysql_list_fields) */
  enum enum_field_types type;   /* Tipo de campo. Veja o mysql_com.h para os tipos */
  unsigned int length;          /* Tamanho da coluna */
  unsigned int max_length;      /* Tamanho máximo do set seleccionado */
  unsigned int flags;           /* Div flags */
  unsigned int decimals;        /* Número de decimais */
} MYSQL_FIELD;
```

my_ulonglong
Este é o tipo usado para o número de linhas e para mysql_affected_rows(), mysql_num_rows(), e mysql_insert_id(). Este tipo fornece um intervalo de 0 a 1.84e19. Nalguns sistemas tentar imprimir o valor do tipo my_ulonglong não trabalhará. Para imprimir tal valor, converta−o para um long sem sinal (unsigned long) e uso o formato de impressão %lu do printf. Example:
%lu printf format. Example:
printf(Number of rows: %lu\n", (unsigned long) mysql_num_rows(result));

typedef unsigned long my_ulonglong;

Ligando−se ao MySQL e fazendo uma consulta

Agora, assumo que tem o MySQL instalado e que está criado um utilizador, uma tabela e uma base de dados. No caso de ter algum problema, por favor dê uma vista de olhos no website www.mysql.com.

Como já dito antes as bibliotecas do MySQL estão na biblioteca mysqlclient. Assim ao compilar o programa MySQL, é necessário adicionar a opção de compilação *−lmysqlclient*. Os ficheiros cabeçalho do MySQL estão sob /usr/include/mysql (tal pode variar da sua distribuição de Linux). O cabeçalho do seu programa pode assemelhar−se a algo como isto:

#include <mysql/mysql.h>

Os tipos de variáveis do MySQL e as funções estão incluídas neste ficheiro cabeçalho.

Depois, precisamos de criar a variável que é usada para se ligar à base de dados. Isto é simplesmente feito com:

MYSQL *mysql;

Antes de se ligar a uma base de dados, devemos iniciar a seguinte função para inicializar a variável mysql:

mysql_init(MYSQL *mysql)

Depois a

```
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);
```

função é usada para se ligar à base de dados. O host é o nome da máquina servidora onde o MySQL está alojado. O user é o utilizador com o qual nos queremos ligar. A passwd é a palavra chave associada ao utilizador. A db é a base de dados à qual nos queremos ligar. A port é o número da porta TCP/IP do servidor MySQL. O unix_socket é o tipo de ligação pretendida. A clientflag é a flag que faz com que o MySQL corra

tipo ODBC. Neste artigo é definida a zero. Esta função retorna zero quando a ligação é estabelecida.

Agora podemos ligarmo−nos a uma base de dados e fazer uma consulta:

```
char *query;
```

Usando esta função podemos construir qualquer sentença em SQL e fazer uma consulta. A função que executa a consulta é:

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

mysql é a variável que usámos acima. O q é a string de consulta em SQL. A length é o tamanho desta string. Se a consulta tiver sucesso sem nenhum erro, a função retorna 0.

Despois de fazer uma consulta necessitamos de uma variável MYSQL_RES no sentido de pudermos usar os valores resultantes da consulta. A seguinte linha cria esta variável:

```
MYSQL_RES *res;
```

Then

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

A função é usada para ler resultados.

Apesar de conseguirmos fazer consultas facilmente, precisamos de algumas outras funções para usar os resultados. A primeira é:

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

Esta função transforma os resultados em linhas. Como notou, a função retorna um tipo disponível de MYSQL_ROW. A seguinte linha cria tal variável:

```
MYSQL_ROW row;
```

Como explicado anteriormente a variável *row* é um vector de strings. Isto significa que a row[0] é a primeira coluna da primeira linha, row[1] é a segunda coluna da primeira linha... Quando usamos mysql_fetch_row, então a variável obtem os dados da próxima linha do resultado. Quando atingimos o fim do resultado então a função retorna um valor negativo. Por fim necessitamos de terminar a ligação:

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

Algumas funções úteis

Vejamos como obter o número de campos de uma tabela. A seguinte função faz isto:

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

Esta função retorna o número de campos de uma tabela.

Para obter o número de linhas do resultado de uma consulta, usamos:

my_ulonglong STDCALL mysql_num_rows(MYSQL_RES *res);

my_ulonglong STDCALL mysql_affected_rows(MYSQL *mysql);

Esta função é usada para saber o número de lihas que são afectadas pelas instruções de INSERT, de DELETE, de UPDATE. Note que a função retorna o tipo my_ulonglong.

Por fim algum código exemplificativo:

```
#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 connectin ot 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);
}
```

Leituras Recomendadas

- O site do MySQL: www.mysql.com
- Documentos fornecidos com o código do MySQL. (Provavelmente sob a directoria /usr/doc)

Translation information:
tr −−> −− : Özcan Güngör <ozcangungor(at)netscape.net>
tr −−> en: Özcan Güngör <ozcangungor(at)netscape.net>
en −−> pt: Bruno Sousa <bruno(at)linuxfocus.org>

2005−01−10, generated by lfparser_pdf version 2.51