编写 MySql 客户端程序需要 libmysqlclient 库和一些头文件，
可通过安装软件包 libmysqlclient15-dev 获得，
然后在/usr/include/mysql 里可以找到 mysql.h
编译：
示例程序参见 mysql_test.c
gcc -o mysql_test mysql_test.c -I/usr/include/mysql/ -lmysqlclient

应用程序与 MySQL 交互时的一般性原则：

1. 通过调用 mysql_library_init()，初始化 MySQL 库，libmysqlclient C 客户端库。

2. 通过调用 mysql_init()初始化连接处理程序，并通过调用 mysql_real_connect()连接到服务器。

3. 发出 SQL 语句并处理其结果——SELECT 的结果存放在一个数据结构里称为结果集，

4. 通过调用 mysql_close()，关闭与 MySQL 服务器的连接。

5. 通过调用 mysql_library_end()，结束 MySQL 库的使用。

程序需要一些 mysql 预定义的数据结构：
需要一个 MYSQL 类型的变量
MYSQL mysql;
需要一个 MYSQL_RES 类型的指针指向的结果集
MYSQL_RES *resultset;

一、初始化库
函数原型：int mysql_library_init(int argc, char **argv, char **groups)
这个函数的参数没什么用，传 0, NULL, NULL 即可。

二、初始化 MYSQL 结构
函数原型：MYSQL *mysql_init(MYSQL *mysql)
MYSQL mysql;
mysql_init(&mysql);
以后的函数都要以这个初始化好的结构为参数。

三、连接 MySQL 服务器

MySQL 服务器是以独立进程的形式存在，它也可能是远程其他起机器上的进程，所以要指定 IP 和端口号。

```
MYSQL *mysql_real_connect(MYSQL *mysql,          //前面初始化过的结构
                          const char *host,      //MySql 服务器所在机器的 IP
                          const char *user,      //MySql 用户名
                          const char *passwd,    //MySql  密码
                          const char *db,  //服务器上的数据库名，一个服务器上可以有许多数据库
                          unsigned int port,     //MySql 服务端口号，传 0 默认端口 3306
                          const char *unix_socket,    // 传 NULL 即可
                          unsigned long client_flag)
```

其中参数 client_flag 一般传 0，如果要支持一次查询传入多条 SQL 语句，这些语句由 ';' 分隔，用 CLIENT_MULTI_STATEMENTS，连接服务器失败返回 NULL

程序中用的参数

```
mysql_real_connect(&mysql, "127.0.0.1", "root", "admin", "test", 0, NULL, CLIENT_MULTI_STATEMENTS))
```

四、执行 SQL 语句

```
int mysql_query(MYSQL *mysql, const char *query)
```

参数 query 就代表 SQL 语句

query 字符串必须以'\0'结尾，不包含 ';'

成功返回 0，出错返回非 0

五、返回结果

SQL 语句分两种，一种是 SELECT 查询，需要返回结果集，另一种是非 SELECT 查询（例如 INSERT、UPDATE、DELETE）可以调用函数发现有多少行被改变。

1）SELECT 查询：

```
MYSQL_RES *mysql_store_result(MYSQL *mysql)
```

此函数返回查询所得的结果集，将查询的全部结果读取到客户端，分配 1 个 MYSQL_RES 结构，并将结果置于该结构中。

如果查询未返回结果集(例如，查询是 INSERT 语句)，mysql_store_result()将返回 NULL 指针。

对于其他查询，不需要调用 mysql_store_result()，但是调用了 mysql_store_result()它也不会导致任何伤害或性能降低，所以通常是调用他然后判断是否是成功执行了 SELECT 或其他查询。

下面的代码段，调用 mysql_store_result(&mysql) 取得结果集，然后将结果集里一行一行的数据打印出来，当用完结果集，或要进行新的查询时，要释放结果集所占内存：

```
resultset = mysql_store_result(&mysql);
if (resultset)  //有结果集返回(resultset 不为空)，说明是 SELECT
{
        //获得结果集中的列数
        int num_fields = mysql_num_fields(resultset);
        //将结果的每一行打印出来
        while((record = mysql_fetch_row(resultset)))
        {
            int i;
            for(i = 0; i < num_fields; i++)
                printf("[%s]", record[i]);
            printf("\n");
        }
        //当用完此结果集(或需要重新查询),
        //释放结果集所占内存
        mysql_free_result(resultset);
 }
```

2）非 SELECT 查询（INSERT、UPDATE、DELETE）

如果调用 mysql_store_result() 会返回 NULL 指针， 这时最好调用

unsigned int mysql_field_count(MYSQL *mysql) 判断查询类型,

如果 mysql_field_count()返回 0 说明执行的查询是 INSERT、UPDATE 或 DELETE 且没有错误。

这时还可以调用 my_ulonglong mysql_affected_rows(MYSQL *mysql) 发现查询所影响的行数。

六、收尾

当不再需要查询时，调用 void mysql_close(MYSQL *mysql) 断开与 MySQL 服务器的连接。