# 概述

# 接口

## MySQL5.7

参考：<https://dev.mysql.com/doc/c-api/5.7/en/>

### Writing C API-Based Client Applications

### C API Basic Interface

#### 连接

##### mysql\_init()

##### mysql\_options()

int mysql\_options(MYSQL \*mysql, enum mysql\_option option, const char \*arg)

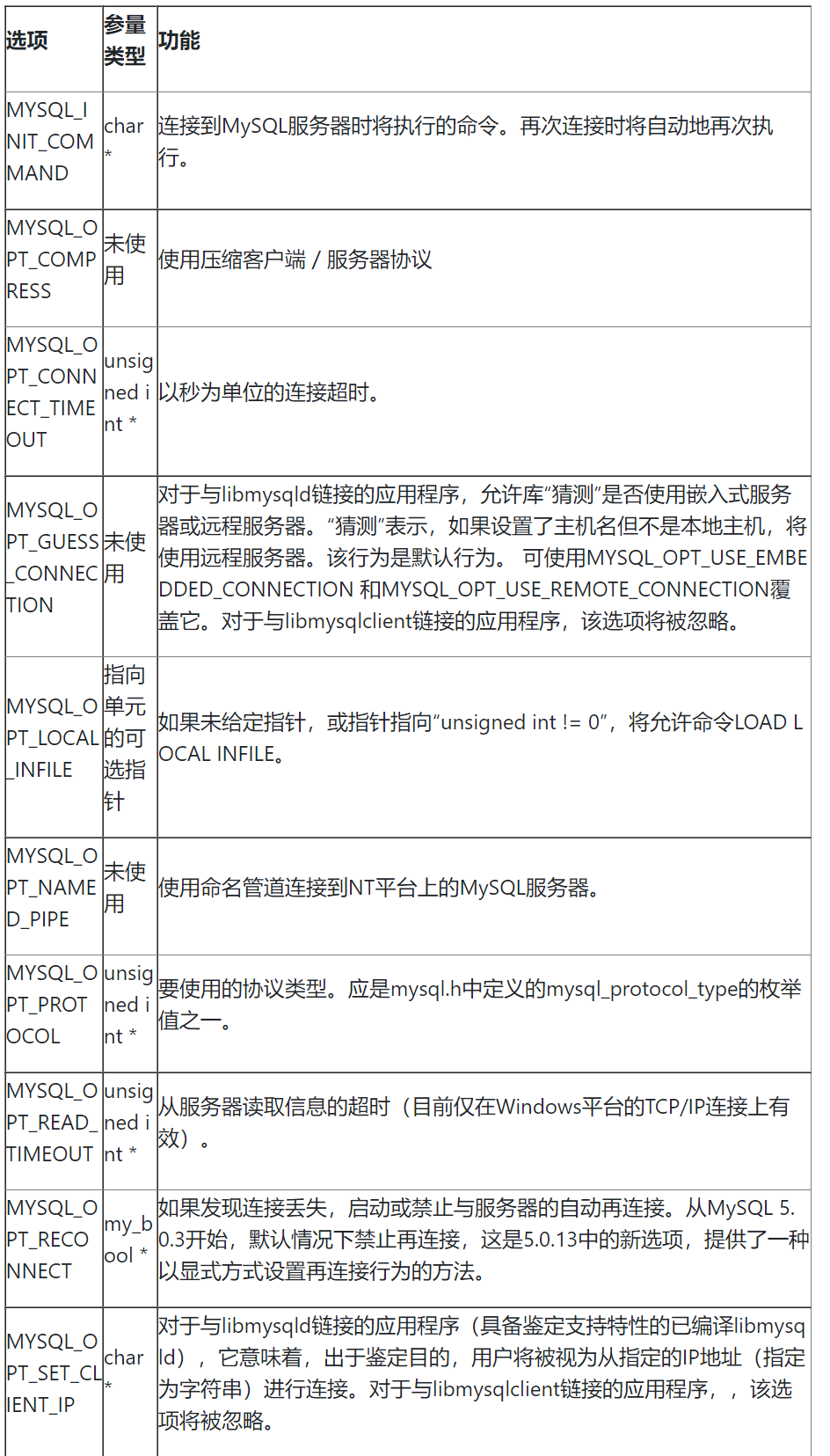
**描述：**

可用于设置额外的连接选项，并影响连接的行为。可多次调用该函数来设置数个选项。

应在mysql\_init()之后、以及mysql\_connect()或mysql\_real\_connect()之前调用mysql\_options()。

选项参量指的是你打算设置的选项。Arg参量是选项的值。如果选项是整数，那么arg应指向整数的值。

**可能的选项值：**





注意，如果使用了MYSQL\_READ\_DEFAULT\_FILE或MYSQL\_READ\_DEFAULT\_GROUP，总会读取客户端组。

选项文件中指定的组可能包含下述选项：



注意，“timeout”（超时）已被“connect-timeout”（连接超时）取代，但为了保持向后兼容，在MySQL 5.1.2-alpha中仍支持“timeout”（超时）。

**返回值：**

成功时返回0。如果使用了未知选项，返回非0值。

##### mysql\_set\_server\_option()

int mysql\_set\_server\_option(MYSQL \*mysql, enum enum\_mysql\_set\_option option)

**描述：**

允许或禁止连接的选项。选项可以取下述值之一：

MYSQL\_OPTION\_MULTI\_STATEMENTS\_ON

允许多语句支持。

MYSQL\_OPTION\_MULTI\_STATEMENTS\_OFF

禁止多语句支持。

**返回值：**

0表示成功，非0值表示出现错误。

**错误：**

CR\_COMMANDS\_OUT\_OF\_SYNC

以不恰当的顺序执行了命令。

CR\_SERVER\_GONE\_ERROR

MySQL服务器不可用。

CR\_SERVER\_LOST

在查询过程中，与服务器的连接丢失。

ER\_UNKNOWN\_COM\_ERROR

服务器不支持mysql\_set\_server\_option()（当服务器版本低于4.1.1时），或服务器不支持试图设置的选项。

##### mysql\_connect()

MYSQL \*mysql\_connect(MYSQL \*mysql, const char \*host, const char \*user, const char \*passwd)

**描述：**

该函数已过时。最好使用mysql\_real\_connect()取而代之。

mysql\_connect()试图建立与运行在主机上的MySQL数据库引擎的连接。在能够执行任何其他API函数之前，mysql\_connect()必须成功完成，但mysql\_get\_client\_info()例外。

这些参数的意义与mysql\_real\_connect()的对应参数的意义相同，差别在于连接参数可以为NULL。在这种情况下，C API将自动为连接结构分配内存，并当调用mysql\_close()时释放分配的内存。该方法的缺点是，如果连接失败，你无法检索错误消息。要想从mysql\_errno()或mysql\_error()获得错误消息，必须提供有效的MYSQL指针。

**返回值：**

与mysql\_real\_connect()的相同。

**错误：**

与mysql\_real\_connect()的相同。

##### mysql\_real\_connect()

MYSQL \*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 long client\_flag)

**描述：**

mysql\_real\_connect()尝试与运行在主机上的MySQL数据库引擎建立连接。在你能够执行需要有效MySQL连接句柄结构的任何其他API函数之前，mysql\_real\_connect()必须成功完成。

参数的指定方式如下：

第1个参数应是已有MYSQL结构的地址。调用mysql\_real\_connect()之前，必须调用mysql\_init()来初始化MYSQL结构。通过mysql\_options()调用，可更改多种连接选项。

“host”的值必须是主机名或IP地址。如果“host”是NULL或字符串"localhost"，连接将被视为与本地主机的连接。如果操作系统支持套接字（Unix）或命名管道（Windows），将使用它们而不是TCP/IP连接到服务器。

“user”参数包含用户的MySQL登录ID。如果“user”是NULL或空字符串""，用户将被视为当前用户。在UNIX环境下，它是当前的登录名。在Windows ODBC下，必须明确指定当前用户名。

“passwd”参数包含用户的密码。如果“passwd”是NULL，仅会对该用户的（拥有1个空密码字段的）用户表中的条目进行匹配检查。这样，数据库管理员就能按特定的方式设置MySQL权限系统，根据用户是否拥有指定的密码，用户将获得不同的权限。

注释：调用mysql\_real\_connect()之前，不要尝试加密密码，密码加密将由客户端API自动处理。

“db”是数据库名称。如果db为NULL，连接会将默认的数据库设为该值。

如果“port”不是0，其值将用作TCP/IP连接的端口号。注意，“host”参数决定了连接的类型。

如果unix\_socket不是NULL，该字符串描述了应使用的套接字或命名管道。注意，“host”参数决定了连接的类型。

client\_flag的值通常为0，但是，也能将其设置为下述标志的组合，以允许特定功能：



对于某些参数，能够从选项文件获得取值，而不是取得mysql\_real\_connect()调用中的确切值。为此，在调用mysql\_real\_connect()之前，应与MYSQL\_READ\_DEFAULT\_FILE或MYSQL\_READ\_DEFAULT\_GROUP选项一起调用mysql\_options()。随后，在mysql\_real\_connect()调用中，为准备从选项文件读取值的每个参数指定“无值”值：

对于host，指定NULL值或空字符串("")。

对于user，指定NULL值或空字符串。

对于passwd，指定NULL值。（对于密码，mysql\_real\_connect()调用中的空字符串的值不能被选项文件中的字符串覆盖，这是因为，空字符串明确指明MySQL账户必须有空密码）。

对于db，指定NULL值或空字符串

对于port，指定“0”值。

对于unix\_socket，指定NULL值。

对于某一参数，如果在选项文件中未发现值，将使用它的默认值，如本节前面介绍的那样。

**返回值：**

如果连接成功，返回MYSQL\*连接句柄。如果连接失败，返回NULL。对于成功的连接，返回值与第1个参数的值相同。

**错误：**

CR\_CONN\_HOST\_ERROR

无法连接到MySQL服务器。

CR\_CONNECTION\_ERROR

无法连接到本地MySQL服务器。

CR\_IPSOCK\_ERROR

无法创建IP套接字。

CR\_OUT\_OF\_MEMORY

内存溢出。

CR\_SOCKET\_CREATE\_ERROR

无法创建Unix套接字。

CR\_UNKNOWN\_HOST

无法找到主机名的IP地址。

CR\_VERSION\_ERROR

协议不匹配，起因于：试图连接到具有特定客户端库（该客户端库使用了不同的协议版本）的服务器。如果使用很早的客户端库来建立与较新的服务器（未使用“--old-protocol”选项开始的）的连接，就会出现该情况。

CR\_NAMEDPIPEOPEN\_ERROR

无法在Windows平台下创建命名管道。

CR\_NAMEDPIPEWAIT\_ERROR

在Windows平台下等待命名管道失败。

CR\_NAMEDPIPESETSTATE\_ERROR

在Windows平台下获取管道处理程序失败。

CR\_SERVER\_LOST

如果connect\_timeout > 0，而且在连接服务器时所用时间长于connect\_timeout秒，或在执行init-command时服务器消失。

##### mysql\_ssl\_set()

##### mysql\_library\_init()

##### mysql\_library\_end()

##### mysql\_query()

**描述：**

执行由“Null终结的字符串”查询指向的SQL查询。正常情况下，字符串必须包含1条SQL语句，而且不应为语句添加终结分号（‘;’）或“\g”。如果允许多语句执行，字符串可包含多条由分号隔开的语句。

mysql\_query()不能用于包含二进制数据的查询，应使用mysql\_real\_query()取而代之（二进制数据可能包含字符‘\0’，mysql\_query()会将该字符解释为查询字符串结束）。

如果希望了解查询是否应返回结果集，可使用mysql\_field\_count()进行检查。

**返回值：**

如果查询成功，返回0。如果出现错误，返回非0值。

**错误：**

CR\_COMMANDS\_OUT\_OF\_SYNC

以不恰当的顺序执行了命令。

CR\_SERVER\_GONE\_ERROR

MySQL服务器不可用。

CR\_SERVER\_LOST

在查询过程中，与服务器的连接丢失。

CR\_UNKNOWN\_ERROR

出现未知错误。

##### mysql\_real\_query()

**描述：**

执行由“query”指向的SQL查询，它应是字符串长度字节“long”。正常情况下，字符串必须包含1条SQL语句，而且不应为语句添加终结分号（‘;’）或“\g”。如果允许多语句执行，字符串可包含由分号隔开的多条语句。

对于包含二进制数据的查询，必须使用mysql\_real\_query()而不是mysql\_query()，这是因为，二进制数据可能会包含‘\0’字符。此外，mysql\_real\_query()比mysql\_query()快，这是因为它不会在查询字符串上调用strlen()。

如果希望知道查询是否应返回结果集，可使用mysql\_field\_count()进行检查。

返回值：

如果查询成功，返回0。如果出现错误，返回非0值。

错误：

CR\_COMMANDS\_OUT\_OF\_SYNC

以不恰当的顺序执行了命令。

CR\_SERVER\_GONE\_ERROR

MySQL服务器不可用。

CR\_SERVER\_LOST

在查询过程中，与服务器的连接丢失。

CR\_UNKNOWN\_ERROR

出现未知错误。

##### mysql\_close()

void mysql\_close(MYSQL \*mysql)

**描述：**

关闭前面打开的连接。如果句柄是由mysql\_init()或mysql\_connect()自动分配的，mysql\_close()还将解除分配由mysql指向的连接句柄。

**返回值：**

无。

**错误：**

无。

##### mysql\_ping()

int mysql\_ping(MYSQL \*mysql)

**描述：**

检查与服务器的连接是否工作。如果连接丢失，将自动尝试再连接。

该函数可被闲置了较长时间的客户端使用，用以检查服务器是否已关闭了连接，并在必要时再次连接。

**返回值：**

如果与服务器的连接有效返回0。如果出现错误，返回非0值。返回的非0值不表示MySQL服务器本身是否已关闭，连接可能因其他原因终端，如网络问题等。

**错误：**

CR\_COMMANDS\_OUT\_OF\_SYNC

以不恰当的顺序执行了命令。

CR\_SERVER\_GONE\_ERROR

MySQL服务器不可用。

CR\_UNKNOWN\_ERROR

出现未知错误。

##### mysql\_shutdown()

int mysql\_shutdown(MYSQL \*mysql, enum enum\_shutdown\_level shutdown\_level)

**描述：**

请求数据库服务器关闭。已连接的用户必须具有SHUTDOWN权限。MySQL 5.1服务器仅支持1种关闭类型，shutdown\_level必须等效于SHUTDOWN\_DEFAULT。设计规划了额外的关闭级别，以便能够选择所需的级别。对于用旧版本libmysqlclient头文件编译并调用mysql\_shutdown()的动态链接可执行程序，需要与旧版的libmysqlclient动态库一起使用。

**返回值：**

0表示成功，非0值表示出现错误。

**错误：**

CR\_COMMANDS\_OUT\_OF\_SYNC

以不恰当的顺序执行了命令。

CR\_SERVER\_GONE\_ERROR

MySQL服务器不可用。

CR\_SERVER\_LOST

在查询过程中，与服务器的连接丢失。

CR\_UNKNOWN\_ERROR

出现未知错误。

#### 属性

##### mysql\_thread\_id()

**描述：**

返回当前连接的线程ID。该值可用作mysql\_kill()的参量以杀死线程。

如果连接丢失，并使用mysql\_ping()进行了再连接，线程ID将改变。这意味着你不应获取线程ID并保存它供以后使用。应在需要时获取它。

**返回值：**

当前连接的线程ID。

##### mysql\_get\_character\_set\_info()

##### mysql\_info()

##### mysql\_get\_client\_info()

##### mysql\_get\_client\_version()

##### mysql\_get\_host\_info()

##### mysql\_get\_proto\_info()

##### mysql\_get\_server\_info()

##### mysql\_get\_server\_version()

##### mysql\_stat()

##### mysql\_sqlstat()

#### DCL

##### mysql\_change\_user()

##### mysql\_character\_set\_name()

##### mysql\_set\_character\_set()

##### mysql\_refresh()

int mysql\_refresh(MYSQL \*mysql, unsigned int options)

描述

该函数用于刷新表或高速缓冲，或复位复制服务器信息。连接的用户必须具有RELOAD权限。

“options”参量是一种位掩码，由下述值的任意组合构成。能够以“Or”（或）方式将多个值组合在一起，用一次调用执行多项操作。

REFRESH\_GRANT

刷新授权表，与FLUSH PRIVILEGES类似。

REFRESH\_LOG

刷新日志，与FLUSH LOGS类似。

REFRESH\_TABLES

刷新表高速缓冲，与FLUSH TABLES类似。

REFRESH\_HOSTS

刷新主机高速缓冲，与FLUSH HOSTS类似。

REFRESH\_STATUS

复位状态变量，与FLUSH STATUS类似。

REFRESH\_THREADS

刷新线程高速缓冲。

REFRESH\_SLAVE

在从复制服务器上，复位主服务器信息，并重新启动从服务器，与RESET SLAVE类似。

REFRESH\_MASTER

在主复制服务器上，删除二进制日志索引中列出的二进制日志文件，并截短索引文件，与RESET MASTER类似。

**返回值：**

0表示成功，非0值表示出现错误。

错误

CR\_COMMANDS\_OUT\_OF\_SYNC

以不恰当的顺序执行了命令。

CR\_SERVER\_GONE\_ERROR

MySQL服务器不可用。

CR\_SERVER\_LOST

在查询过程中，与服务器的连接丢失。

CR\_UNKNOWN\_ERROR

出现未知错误。

##### mysql\_reload()

int mysql\_reload(MYSQL \*mysql)

描述

请求MySQL服务器重新加载授权表。连接的用户必须具有RELOAD权限。

该函数已过时。最好使用mysql\_query()来发出SQL FLUSH PRIVILEGES语句。

返回值

0表示成功，非0值表示出现错误。

错误

CR\_COMMANDS\_OUT\_OF\_SYNC

以不恰当的顺序执行了命令。

CR\_SERVER\_GONE\_ERROR

MySQL服务器不可用。

CR\_SERVER\_LOST

在查询过程中，与服务器的连接丢失。

CR\_UNKNOWN\_ERROR

出现未知错误。

##### mysql\_kill()

#### Transaction

##### mysql\_autocommit()

##### mysql\_commit()

##### mysql\_rollback()

my\_bool mysql\_rollback(MYSQL \*mysql)

描述

回滚当前事务。

该函数的动作取决于completion\_type系统变量的值。尤其是，如果completion\_type的值为“2”，终结事务后，服务器将执行释放操作，并关闭客户端连接。客户端程序应调用mysql\_close()，从客户端一侧关闭连接。

返回值

如果成功，返回0，如果出现错误，返回非0值。

错误

无。

#### SQL

##### mysql\_create\_db()

int mysql\_create\_db(MYSQL \*mysql, const char \*db)

描述

创建由db参数命名的数据库。

该函数已过时。最好使用mysql\_query()来发出SQL CREATE DATABASE语句。

返回值

如果数据库已成功创建，返回0，如果出现错误，返回非0值。

错误

CR\_COMMANDS\_OUT\_OF\_SYNC

以不恰当的顺序执行了命令。

CR\_SERVER\_GONE\_ERROR

MySQL服务器不可用。

CR\_SERVER\_LOST

在查询过程中，与服务器的连接丢失。

CR\_UNKNOWN\_ERROR

出现未知错误。

##### mysql\_drop\_db()

##### mysql\_select\_db()

##### mysql\_list\_dbs()

##### mysql\_list\_fields()

##### mysql\_list\_processes()

##### mysql\_list\_tables()

##### mysql\_insert\_id()

#### DEBUG

##### mysql\_debug()

##### mysql\_dump\_debug\_info()

#### 数据解析

##### mysql\_hex\_string()

##### mysql\_escape\_string()

应使用mysql\_real\_escape\_string()取而代之！

该函数与mysql\_real\_escape\_string()等同，但mysql\_real\_escape\_string()会将连接处理程序作为其第1个参量，并按照当前字符集对字符串进行转义处理。mysql\_escape\_string()不采用连接参量，而且不考虑当前字符集设置。

##### mysql\_real\_escape\_string()

unsigned long mysql\_real\_escape\_string(MYSQL \*mysql, char \*to, const char \*from, unsigned long length)

注意，mysql必须是有效的开放式连接。之所以需要它是因为，转义功能取决于服务器使用的字符集。

**描述：**

该函数用于创建可在SQL语句中使用的合法SQL字符串。

按照连接的当前字符集，将“from”中的字符串编码为转义SQL字符串。将结果置于“to”中，并添加1个终结用NULL字节。编码的字符为NUL (ASCII 0)、‘\n’、‘\r’、‘\’、‘'’、‘"’、以及Control-Z（请参见9.1节，“文字值”）。（严格地讲，MySQL仅需要反斜杠和引号字符，用于引用转义查询中的字符串。该函数能引用其他字符，从而使得它们在日志文件中具有更好的可读性）。

“from”指向的字符串必须是长度字节“long”。必须为“to”缓冲区分配至少length\*2+1字节。在最坏的情况下，每个字符或许需要使用2个字节进行编码，而且还需要终结Null字节。当mysql\_real\_escape\_string()返回时，“to”的内容是由Null终结的字符串。返回值是编码字符串的长度，不包括终结用Null字符。

如果需要更改连接的字符集，应使用mysql\_set\_character\_set()函数，而不是执行SET NAMES (或SET CHARACTER SET)语句。mysql\_set\_character\_set()的工作方式类似于SET NAMES，但它还能影响mysql\_real\_escape\_string()所使用的字符集，而SET NAMES则不能。

##### mysql\_store\_result()

对于成功检索了数据的每个查询（SELECT、SHOW、DESCRIBE、EXPLAIN、CHECK TABLE等），必须调用mysql\_store\_result()或mysql\_use\_result()。

对于其他查询，不需要调用mysql\_store\_result()或mysql\_use\_result()，但是如果在任何情况下均调用了mysql\_store\_result()，它也不会导致任何伤害或性能降低。通过检查mysql\_store\_result()是否返回0，可检测查询是否没有结果集（以后会更多）。

如果希望了解查询是否应返回结果集，可使用mysql\_field\_count()进行检查。

mysql\_store\_result()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，mysql\_store\_result()将返回Null指针（例如，如果查询是INSERT语句）。

如果读取结果集失败，mysql\_store\_result()还会返回Null指针。通过检查mysql\_error()是否返回非空字符串，mysql\_errno()是否返回非0值，或mysql\_field\_count()是否返回0，可以检查是否出现了错误。

如果未返回行，将返回空的结果集。（空结果集设置不同于作为返回值的空指针）。

一旦调用了mysql\_store\_result()并获得了不是Null指针的结果，可调用mysql\_num\_rows()来找出结果集中的行数。

可以调用mysql\_fetch\_row()来获取结果集中的行，或调用mysql\_row\_seek()和mysql\_row\_tell()来获取或设置结果集中的当前行位置。

一旦完成了对结果集的操作，必须调用mysql\_free\_result()。

##### mysql\_free\_result()

##### mysql\_use\_result()

对于成功检索数据的每个查询（SELECT、SHOW、DESCRIBE、EXPLAIN），必须调用mysql\_store\_result()或mysql\_use\_result()。

mysql\_use\_result()将初始化结果集检索，但并不像mysql\_store\_result()那样将结果集实际读取到客户端。它必须通过对mysql\_fetch\_row()的调用，对每一行分别进行检索。这将直接从服务器读取结果，而不会将其保存在临时表或本地缓冲区内，与mysql\_store\_result()相比，速度更快而且使用的内存也更少。客户端仅为当前行和通信缓冲区分配内存，分配的内存可增加到max\_allowed\_packet字节。

另一方面，如果你正在客户端一侧为各行进行大量的处理操作，或者将输出发送到了用户可能会键入“^S”（停止滚动）的屏幕，就不应使用mysql\_use\_result()。这会绑定服务器，并阻止其他线程更新任何表（数据从这类表获得）。

使用mysql\_use\_result()时，必须执行mysql\_fetch\_row()，直至返回NULL值，否则，未获取的行将作为下一个检索的一部分返回。C API给出命令不同步错误，如果忘记了执行该操作，将不能运行该命令。

不应与从mysql\_use\_result()返回的结果一起使用mysql\_data\_seek()、mysql\_row\_seek()、mysql\_row\_tell()、mysql\_num\_rows()或mysql\_affected\_rows()，也不应发出其他查询，直至mysql\_use\_result()完成为止。（但是，提取了所有行后，mysql\_num\_rows()将准确返回提取的行数）。

一旦完成了对结果集的操作，必须调用mysql\_free\_result()。

使用libmysqld嵌入式服务器时，由于在调用mysql\_free\_result()之前，内存使用将随着每个检索的行增加，内存效益将基本丧失。

**返回值：**

MYSQL\_RES结果结构。如果出现错误，返回NULL。

**错误：**

如果成功，mysql\_use\_result()将复位mysql\_error()和mysql\_errno()。

CR\_COMMANDS\_OUT\_OF\_SYNC

以不恰当的顺序执行了命令。

CR\_OUT\_OF\_MEMORY

内存溢出。

CR\_SERVER\_GONE\_ERROR

MySQL服务器不可用。

CR\_SERVER\_LOST

在查询过程中，与服务器的连接丢失。

CR\_UNKNOWN\_ERROR

出现未知错误。

##### mysql\_more\_results()

##### mysql\_next\_result()

##### mysql\_data\_seek()

void mysql\_data\_seek(MYSQL\_RES \*result, my\_ulonglong offset)

描述

在查询结果集中寻找任意行。偏移值为行号，范围从0到mysql\_num\_rows(result)-1。

该函数要求结果集结构包含查询的所有结果，因此，so mysql\_data\_seek()仅应与mysql\_store\_result()联合使用，而不是与mysql\_use\_result()。

返回值

无。

错误

无。

##### mysql\_field\_count()

##### mysql\_field\_seek()

##### mysql\_field\_tell()

##### mysql\_fetch\_field()

##### mysql\_fetch\_field\_direct()

##### mysql\_fetch\_row()

##### mysql\_fetch\_fields()

##### mysql\_fetch\_lengths()

##### mysql\_row\_seek()

MYSQL\_ROW\_OFFSET mysql\_row\_seek(MYSQL\_RES \*result, MYSQL\_ROW\_OFFSET offset)

描述

将行光标置于查询结果集中的任意行。“offset”值是行偏移量，它应是从mysql\_row\_tell()或mysql\_row\_seek()返回的值。该值不是行编号，如果你打算按编号查找结果集中的行，请使用mysql\_data\_seek()。

该函数要求在结果集的结构中包含查询的全部结果，因此，mysql\_row\_seek()仅应与mysql\_store\_result()一起使用，而不是与mysql\_use\_result()。

返回值

行光标的前一个值。该值可传递给对mysql\_row\_seek()的后续调用。

错误

无。

##### mysql\_row\_tell()

MYSQL\_ROW\_OFFSET mysql\_row\_tell(MYSQL\_RES \*result)

描述

对于上一个mysql\_fetch\_row()，返回行光标的当前位置。该值可用作mysql\_row\_seek()的参量。

仅应在mysql\_store\_result()之后，而不是mysql\_use\_result()之后使用mysql\_row\_tell()。

返回值

行光标的当前偏移量。

错误

无。

##### mysql\_num\_fields()

unsigned int mysql\_num\_fields(MYSQL\_RES \*result)

要想传递MYSQL\*参量取而代之，请使用无符号整数mysql\_field\_count(MYSQL \*mysql)。

描述

返回结果集中的行数。

注意，你可以从指向结果集的指针或指向连接句柄的指针获得行数。如果mysql\_store\_result()或mysql\_use\_result()返回NULL，应使用连接句柄（因而没有结果集指针）。在该情况下，可调用mysql\_field\_count()来判断mysql\_store\_result()是否生成了非空结果。这样，客户端程序就能采取恰当的行动，而不需要知道查询是否是SELECT语句（或类似SELECT的语句）。在下面的示例中，介绍了执行该操作的方式。

返回值

表示结果集中行数的无符号整数。

错误

无。

##### mysql\_num\_rows()

my\_ulonglong mysql\_num\_rows(MYSQL\_RES \*result)

描述

返回结果集中的行数。

mysql\_num\_rows()的使用取决于是否采用了mysql\_store\_result()或mysql\_use\_result()来返回结果集。如果使用了mysql\_store\_result()，可以立刻调用mysql\_num\_rows()。如果使用了mysql\_use\_result()，mysql\_num\_rows()不返回正确的值，直至检索了结果集中的所有行为止。

返回值

结果集中的行数。

错误

无。

##### mysql\_affected\_rows()

my\_ulonglong mysql\_affected\_rows(MYSQL \*mysql)

**描述：**

返回上次UPDATE更改的行数，上次DELETE删除的行数，或上次INSERT语句插入的行数。对于UPDATE、DELETE或INSERT语句，可在mysql\_query()后立刻调用。对于SELECT语句，mysql\_affected\_rows()的工作方式与mysql\_num\_rows()类似。

**返回值：**

大于0的整数表明受影响或检索的行数。“0”表示UPDATE语句未更新记录，在查询中没有与WHERE匹配的行，或未执行查询。“-1”表示查询返回错误，或者，对于SELECT查询，在调用mysql\_store\_result()之前调用了mysql\_affected\_rows()。由于mysql\_affected\_rows()返回无符号值，通过比较返回值和“(my\_ulonglong)-1”或等效的“(my\_ulonglong)~0”，检查是否为“-1”。

**错误：**

无。

#### ERROR

##### mysql\_warning\_count()

返回执行前一个SQL语句期间生成的告警数目。

返回值：

告警计数。

##### mysql\_eof()

my\_bool mysql\_eof(MYSQL\_RES \*result)

描述

该函数已过时。应使用mysql\_errno()或mysql\_error()取而代之。

mysql\_eof()确定是否已读取了结果集的最后1行。

如果通过成功调用mysql\_store\_result()获得了结果集，客户端将在1次操作中收到整个结果集。在该情况下，从mysql\_fetch\_row()返回的NULL总表示已到达结果集末尾，而且没必要调用mysql\_eof()。与mysql\_store\_result()一起使用时，mysql\_eof()总返回“真”。

另一方面，如果你使用mysql\_use\_result()来初始化结果集检索，当重复调用mysql\_fetch\_row()时，将逐个地从服务器获取结果集的行。由于在该过程中，可能出现连接上的错误，从mysql\_fetch\_row()返回的NULL值不一定表示已正常地抵达结果集末尾。在该情况下，可以使用mysql\_eof()来判定出现了什么情况。如果抵达结果集末尾，mysql\_eof()返回非0值，如果出现错误，返回0。

从历史的角度上看，mysql\_eof()在日期上早于标准的MySQL错误函数mysql\_errno()和mysql\_error()。由于这类错误函数提供了相同的信息，它们优先于已过时的mysql\_eof()。事实上，它们提供了更多信息，这是因为，mysql\_eof()仅返回布尔值，错误函数能够在出现错误时指明错误的原因。

返回值

如果未出现错误，返回0。如果抵达结果集的末尾，返回非0值。

错误

无。

##### mysql\_errno()

##### mysql\_error()

### C API Prepared Statement Interface

### C API Thread Interface

#### my\_init()

my\_init初始化MySQL需要的全局变量，它通常调用mysql\_thread\_init用于初始化线程。

my\_init()函数必须在程序初始化MySQL库的阶段先被调用。而且，在mysql\_init(),mysql\_library\_init(),mysql\_server\_init()和mysql\_connect()函数中自动被调用，所以如果已经调用了这些函数，则没有必要再次调用my\_init()函数。

头文件：

#include <my\_global.h>

#include <my\_sys.h>

返回值：无

#### mysql\_thread\_end()

该函数在pthread\_exit()函数之前调用，用来释放mysql\_thread\_init()分配的内存。

mysql\_thread\_end()不会被client library自动调用，所以该函数必须手动调用防止内存泄漏。

#### mysql\_thread\_init()

该函数在创建thread之后需要最先被调用以初始化thread-specific的值。

该函数被my\_init()自动调用，所以my\_init()被调用了则没必要再调用mysql\_thread\_init()。

返回值：

1. 成功；
2. 非0-出错

#### mysql\_thread\_safe()

该函数判断client library是否是thread-safe编译的。

返回值：

1-client library是thread-safe；

0-其他

### C API Embedded Server Functions

参考：<http://bcxw.net/api/mysql/libmysqld-overview.html>

使用嵌入式MySQL服务器库，能够在客户端应用程序中使用具备全部特性的MySQL服务器。主要优点在于，增加了速度，并使得嵌入式应用程序的管理更简单。

嵌入式服务器库是以MySQL的客户端／服务器版本为基础的，采用C/C++语言编写。其结果是嵌入式服务器也是用C/C++语言编写的。在其他语言中，嵌入式服务器不可用。

#### mysql\_library\_init()

<https://www.csdn.net/tags/NtDacgysODI1ODItYmxvZwO0O0OO0O0O.html>

<https://blog.csdn.net/hjl240/article/details/75578004>

应用程序使用MySQL有如下几个步骤：

1、通过调用mysql\_library\_init()来初始化MySQL库。这个函数同时存在于libmysqlclient 客户端库和libmysqld服务器端库，因此当你建立一个正常的客户端程序时要链接-libmysqlclient，或者建立一个服务器应用程序时，要链接 -libmysqld。

2、通过调用mysql\_init()来初始化一个连接句柄。通过调用mysql\_real\_connect()来连接服务器。

3、使用SQL语句操作数据库，并处理这些结果。

4、通过调用mysql\_close()来关闭MySQL服务器的连接。

5、通过调用mysql\_library\_end()来终止使用MySQL库。

#### mysql\_library\_end()

#### mysql\_server\_init()

#### mysql\_server\_end()

### C API Client Plugin Interface

## MySQL8.0

参考：<https://dev.mysql.com/doc/c-api/8.0/en/>