MySql接口

mysql_init()

```
MYSQL *mysql_init(MYSQL *mysql)
```

说明

分配或初始化适合mysql_real_connect()的`MYSQL` object。如果`mysql`是`NULL`指针，则 function 分配，初始化并返回一个新的 object。否则，初始化 object 并返回 object 的地址。如果mysql_init()分配一个新的 object，则在调用mysql_close()来关闭连接时将释放它。

在非多线程环境中，mysql_init()会根据需要自动调用mysqllibrary_init()。但是，mysqllibrary_init()在多线程环境中不是 thread-safe，因此也不是mysql_init()。在调用mysql_init()之前，要么在产生任何线程之前调用mysqllibrary_init()，要么使用 mutex 来保护mysqllibrary_init()调用。这应该在任何其他 client library 调用之前完成。

Return 值

初始化的`MYSQL*`处理程序。 `NULL`如果没有足够的 memory 来分配新的 object。

错误

如果 memory 不足，则返回`NULL`。

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)
```

说明

mysql_real_connect()尝试在`host`上建立与 MySQL 数据库引擎的连接 running。必须先成功完成mysql_real_connect()才能执行需要有效`MYSQL`连接处理程序结构的任何其他 API 函数。

例子

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"your_prog_name");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
    fprintf(stderr, "Failed to connect to database: Error: %s\n",
          mysql_error(&mysql));
}
```

mysql_real_query()

```
int mysql_real_query(MYSQL *mysql, const char *stmt_str, unsigned long length)
```

说明

mysql_real_query()执行`stmt_str`指向的 SQL 语句，string `length` bytes long。通常，string 必须由单个 SQL 语句组成，不带终止分号(`;`)或`\g`。如果启用了 multiple-statement 执行，则 string 可以包含由分号分隔的多个 statement。见第 27.8.16 节，“C API 多语句执行支持”。

mysql_query()不能用于包含二进制数据的 statements;你必须改用mysql_real_query()。 (二进制数据可能包含`\0`字符，mysql_query()解释为语句的结尾 string.)此外，mysql_real_query()比mysql_query()更快，因为它不会在语句 string 上调用`strlen()`。

如果您想知道语句是否返回结果集，可以使用mysql_field_count()来检查它。见第 27.8.7.22 节，“mysql_field_count()”。

Return 值

成功归零。如果发生错误，则为非零。

错误

- CR_COMMANDS_OUT_OF_SYNC

命令是在一个不正确的 order 中执行的。

- CR_SERVER_GONE_ERROR

MySQL 服务器已经消失了。

- CR_SERVER_LOST

查询期间丢失了与服务器的连接。

- CR_UNKNOWN_ERROR

出现未知错误。

mysql_affected_rows()

```
my_ulonglong mysql_affected_rows(MYSQL *mysql)
```

说明

在使用mysql_query()或mysql_real_query()执行语句后，可以立即调用mysql_affected_rows()。它返回最后一个语句更改，删除或插入的行数(如果它是UPDATE，删除或插入)。对于选择 statements，mysql_affected_rows()的作用类似于mysql_num_rows()。

Return 值

大于零的整数表示受影响或检索的行数。零表示没有为UPDATE语句更新记录，没有与查询中的`WHERE`子句匹配的行或者尚未执行任何查询。 -1 表示查询返回了错误，或者对于选择查询，mysql_affected_rows()在调用mysqlstore_result()之前被调用。

因为mysql_affected_rows()返回无符号 value，所以可以通过将 return value 与`(my_ulonglong)-1`(或等同于`(my_ulonglong)~0`)进行比较来检查-1。

错误

没有。

例子

```
char *stmt = "UPDATE products SET cost=cost*1.25
              WHERE group=10";
mysql_query(&mysql,stmt);
printf("%ld products updated",
       (long) mysql_affected_rows(&mysql));
```

mysqlstore_result()

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

说明

在调用mysql_query()或mysql_real_query()之后，必须为成功生成结果集的每个语句调用mysqlstore_result()或mysql_use_result()(选择，节目，描述，说明，检查 TABLE等)。完成结果集后，还必须调用mysql_free_result()。

Return 值

指向具有结果的`MYSQL_RES`结果结构的指针。 `NULL`如果语句没有 return 结果集或发生错误。要确定是否发生错误，请检查mysql_error()是否返回非空 string，mysql_errno()是否返回非零，或mysql_field_count()是否返回零。

错误

如果成功，mysqlstore_result()重置mysql_error()和mysql_errno()。

mysql_num_fields()

```
unsigned int mysql_num_fields(MYSQL_RES *result)
```

要改为传递`MYSQL*`参数，请使用`unsigned int mysql_field_count(MYSQL *mysql)`。

说明

返回结果集中的列数。

mysql_fetch_field()

```
MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *result)
```

说明

返回结果集的一列定义为`MYSQL_FIELD`结构。重复调用此函数以检索有关结果集中所有列的信息。当没有剩下的字段时，mysql_fetch_field()返回`NULL`。

Return 值

当前列的`MYSQL_FIELD`结构。如果没有列，则`NULL`。

mysql_fetch_row()

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

说明

mysql_fetch_row()检索结果集的下一行：

- 在mysqlstore_result()之后使用时，如果没有要检索的行，mysql_fetch_row()将返回`NULL`。
- 在mysql_use_result()之后使用时，如果没有更多行要检索或发生错误，则mysql_fetch_row()返回`NULL`。

行中的值数由mysql_num_fields(result)给出。如果`row`在调用mysql_fetch_row()时保持 return value，则指向值的指针将被`row[0]`到`row[mysql_num_fields(result)-1]`访问。行中的`NULL`值由`NULL`指针指示。

可以通过调用mysql_fetch_lengths()来获得行中字段值的长度。包含`NULL`的空字段和字段都具有长度 0;你可以通过检查字段 value 的指针来区分它们。如果指针是`NULL`，则该字段为`NULL`;否则，该字段为空。

Return 值

下一行的`MYSQL_ROW`结构，或`NULL`。 `NULL` return 的含义取决于在mysql_fetch_row()之前调用的function：

- 在mysqlstore_result()之后使用时，如果没有要检索的行，mysql_fetch_row()将返回`NULL`。
- 在mysql_use_result()之后使用时，如果没有要检索的行或发生错误，则mysql_fetch_row()返回`NULL`。要确定是否发生错误，请检查mysql_error()是否返回非空 string 或mysql_errno()是否返回非零。

例子

```
MYSQL_ROW row;
unsigned int num_fields;
unsigned int i;

num_fields = mysql_num_fields(result);
while ((row = mysql_fetch_row(result)))
{
   unsigned long *lengths;
   lengths = mysql_fetch_lengths(result);
   for(i = 0; i < num_fields; i++)
   {
       printf("[%.*s] ", (int) lengths[i],
              row[i] ? row[i] : "NULL");
   }
   printf("\n");
}
```