**教辅：连接数据库**

**一、连接例程**

用C语言连接MySQL数据库包含两个步骤

1）初始化一个连接句柄结构

**#include “mysql.h”**

**MYSQL \*mysql\_init(MYSQL \*);**

作用：初始化连接句柄

返回值：通常传递NULL给这个例程，它会返回一个指向新分配的连接句柄结构的指针

若传递一个已有的结构，它将被重新初始化。

出错时返回NULL

2）实际进行连接

**#include “mysql.h”**

**MYSQL \*mysql\_real\_connect(MYSQL \*connection,**

**const char \*server\_host,**

**const char \*sql\_user\_name,**

**const char \*sql\_password,**

**const char \*db\_name,**

**unsigned int port\_number,**

**const char \*unix\_socket\_name,**

**unsigned int flags);**

作用：进行实际连接

参数：connection 必须指向已经被mysql\_init初始化过的结构

Sql\_user\_name和sql\_password如果登录名为NULL，则假设登录名为当前Linux用户的登录ID，

如果密码为NULL，将只能访问服务器上无需密码就可访问的数据，密码会在

通过网络传输前进行加密

Port\_number和unix\_socket\_name应该分别为0和NULL

flags用来对一些定义的位模式进行OR操作

返回值：若无法连接，它将返回NULL，mysql\_error函数可以提供有帮助的信息

**3）关闭连接**

**#include “mysql.h”**

**Void mysql\_close(MYSQL \*connection);**

如果连接是由mysql\_init建立的，MYSQL结构会被释放。指针将会失效并无法再次使用

**4）、mysql\_options例程**

可以在mysql\_init和mysql\_real\_connect之间调用设置一些选项

**#include “mysql.h”**

**Int mysql\_options(MYSQL \*connection, enum option\_to\_set, const char \*argument);**

说明：mysql\_options例程一次只能设置一个选项，所以每设置一个选项就得调用它一次，同时，该例程必须出现在mysql\_init和mysql\_real\_connect之间即可。

选项

|  |  |  |
| --- | --- | --- |
| enum选项 | 实际参数类型 | 说明 |
| MySQL\_OPT\_CONNECT\_TIMEOUT | Const unsigned int \* | 连接超时之前的等待秒数 |
| MySQL\_OPT\_COMPRESS | None, 使用NULL | 网络连接中使用压缩机制 |
| MySQL\_INIT\_COMMAND | Const char \* | 每次连接建立后发送的命令 |

返回值：一次成功的调用将返回0。因为它仅仅是用来设置标志，所以失败总是意味着使用了一个无效的选项。

例：如果要设置连接超时时间为7秒，其代码片断如下：

Unsigned int timeout = 7;

…

connection=mysql\_init(NULL);

ret=mysql\_option(connection, MYSQL\_OPT\_CONNECTION\_TIMEOUT, (const char \*)&timeout);

if (ret){

/Handle error \*/

…

}

connection=mysql\_real\_connect(connection …

**二、错误处理**

MySQL使用由连接句柄结构报告的返回码，两个必备的例程是：

**Unsigned int mysql\_errno(MYSQL \*connection); 返回错误码**

**和**

**char \*mysql\_error(MYSQL \*connection); 返回错误码对应的文本信息**

可以通过调用mysql\_errno并传递连接结构来获得错误码（错误码在头文件errmsg.h（报告客户端错误）或mysqld\_error.h（报告服务器端错误）中定义），错误码通常都是非0值。如果未设定错误码，它将返回0.

每次调用库都会更新错误码，所以只能得到最后一个执行命令的错误码。

**三、执行SQL语句**

**1、执行SQL语句的主要API函数**

**Int mysql\_query(MYSQL \*connection, const char \*query);**

说明：此例程接受连接结构指针和文本字符串形式的有效SQL语句

返回值：如果成功，返回0

**2、检查受影响的行数**

**My\_ulonglong mysql\_affected\_rows(MYSQL \*connection);**

说明：返回值的数据类型是my\_ulonglong（无符号长整型），使用printf时，应使用的转换符为%lu

作用：函数返回受update, insert或delete查询影响的行数

返回值：0 没有受影响的行数

正数受语句影响的行数

**四、例：更新数据---update1.c**

1 #include <stdlib.h>

2 #include <stdio.h>

3

4 #include "mysql.h"

5

6 int main(int argc, char \*argv[]) {

7 MYSQL my\_connection;

8 int res;

9

10 mysql\_init(&my\_connection);

11 if (mysql\_real\_connect(&my\_connection, "localhost",

12 "username", "secret", "foo", 0, NULL, 0)) {

13 printf("Connection success\n");

15 res = mysql\_query(&my\_connection, "UPDATE children SET AGE = 4 WHERE fname = 'Ann'");

16 if (!res) {

17 printf("Updated %lu rows\n", (unsigned long)mysql\_affected\_rows(&my\_connection));

18 } else {

19 fprintf(stderr, "Update error %d: %s\n", mysql\_errno(&my\_connection),

mysql\_error(&my\_connection));

21 }

22

23 mysql\_close(&my\_connection);

24 } else {

25 fprintf(stderr, "Connection failed\n");

26 if (mysql\_errno(&my\_connection)) {

27 fprintf(stderr, "Connection error %d: %s\n", mysql\_errno(&my\_connection),

mysql\_error(&my\_connection));

29 }

30 }

31

32 return EXIT\_SUCCESS;

33 }

**四、编译：# gcc -I/usr/include/mysql program.c -L/usr/lib/mysql -lmysqlclient -o program**

其中：-I(大写的i) 指定include路径

-L(大写的L) 指定库文件路径

-l(小写的l) 指定链接的库模块