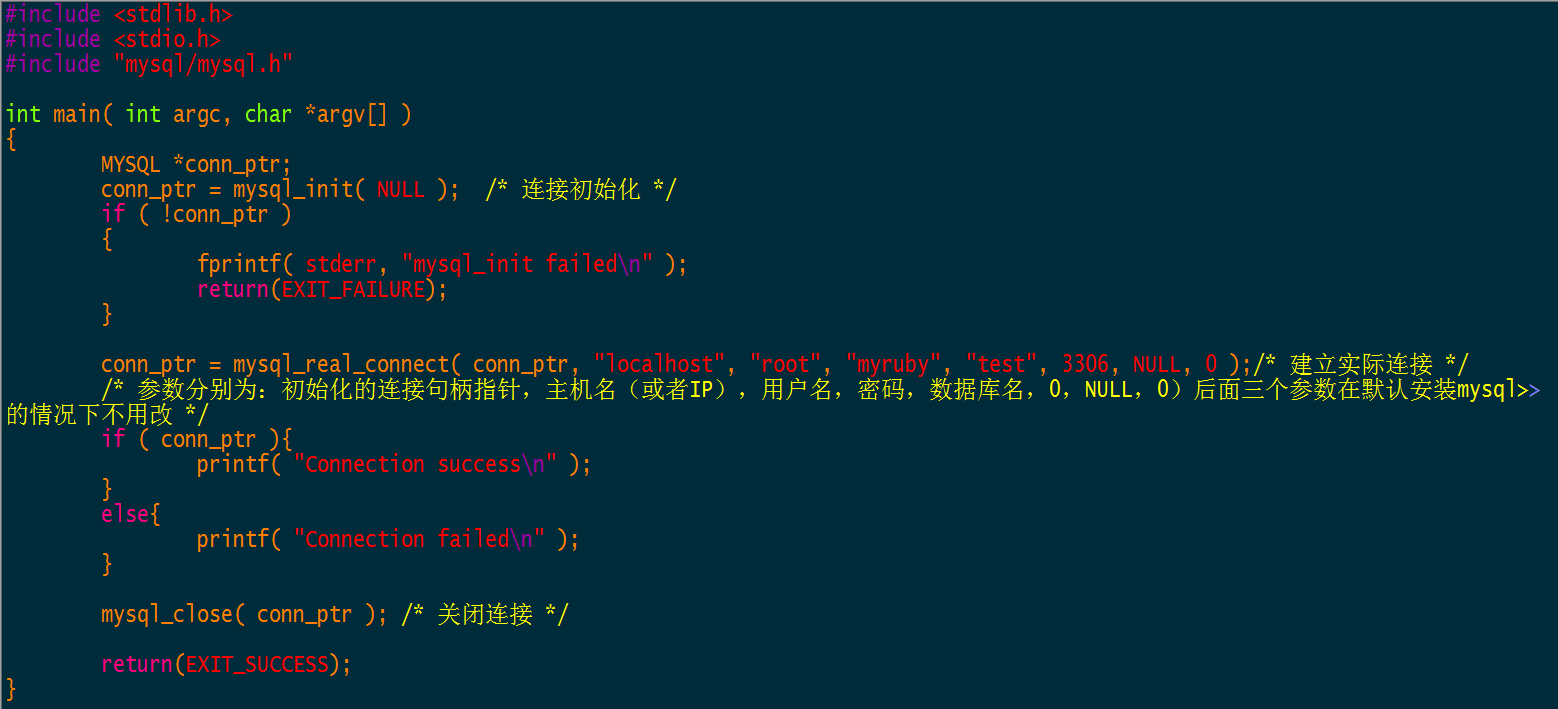
**设计说明文档**

（1）设计说明

① 先看一个简单的连接数据库程序 test\_mysql.c



保存之后 使用如下命令进行编译 （涉及后续编译原理课程）



接下来我们执行生成的文件



显示如上 即成功连接了mysql数据库。

② 连接并操作数据库

首先我们在linux下 登录mysql

以下命令简单的演示了创建数据库的过程，数据名为 RUNOOB:

[root@host]# mysql -u root -p

Enter password:\*\*\*\*\*\* # 登录后进入终端

mysql> create DATABASE RUNOOB;

以下例子中我们将在 RUNOOB 数据库中创建数据表runoob\_tbl：

CREATE TABLE IF NOT EXISTS `runoob\_tbl`(

`runoob\_id` INT UNSIGNED AUTO\_INCREMENT,

`runoob\_title` VARCHAR(100) NOT NULL,

`runoob\_author` VARCHAR(40) NOT NULL,

`submission\_date` DATE,

PRIMARY KEY ( `runoob\_id` )

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

**通过命令提示符创建表**

通过 mysql> 命令窗口可以很简单的创建MySQL数据表。你可以使用 SQL 语句 **CREATE TABLE** 来创建数据表。

**实例**

以下为创建数据表 runoob\_tbl 实例:

root@host# mysql -u root -p

Enter password:\*\*\*\*\*\*\*

mysql> use RUNOOB;

Database changed

mysql> CREATE TABLE runoob\_tbl(

-> runoob\_id INT NOT NULL AUTO\_INCREMENT,

-> runoob\_title VARCHAR(100) NOT NULL,

-> runoob\_author VARCHAR(40) NOT NULL,

-> submission\_date DATE,

-> PRIMARY KEY ( runoob\_id )

-> )ENGINE=InnoDB DEFAULT CHARSET=utf8;

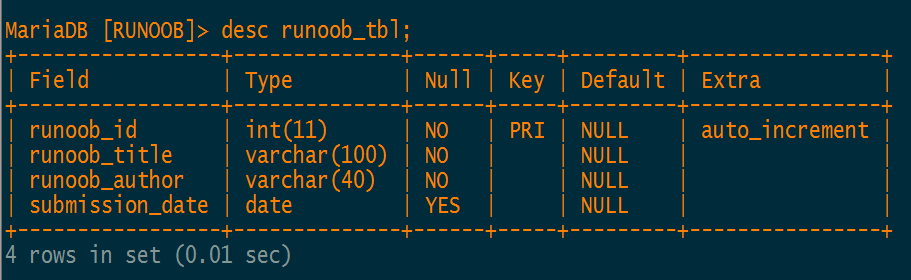
Query OK, 0 rows affected (0.16 sec)

mysql>

**注意：**MySQL命令终止符为分号 **;** 。

**注意：** **->** 是换行符标识，不要复制。

执行成功后，就可以通过命令行查看表结构：



③ 使用C代码来插入两条数据

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "mysql/mysql.h"

int main()

{

MYSQL \*conn = mysql\_init(NULL); //初始化服务器句柄

/\*登陆服务器\*/

if(!mysql\_real\_connect(conn, "localhost", "root", "myruby", "RUNOOB", 3306, NULL, 0))

{

fprintf(stderr, "mysql\_real\_connect: %s\n", mysql\_error(conn));

return -1;

}

MYSQL\_STMT \*stmt = mysql\_stmt\_init(conn); //创建MYSQL\_STMT句柄

char \*query = "insert into runoob\_tbl values(?, ?, ?, ?);";

if(mysql\_stmt\_prepare(stmt, query, strlen(query)))

{

fprintf(stderr, "mysql\_stmt\_prepare: %s\n", mysql\_error(conn));

return -1;

}

printf("id title author date: ");

int id;

char title[20];

char author[20];

char date[20];

scanf("%d %s %s %s", &id, title, author, date);

MYSQL\_BIND params[4];

memset(params, 0, sizeof(params));

params[0].buffer\_type = MYSQL\_TYPE\_LONG;

params[0].buffer = &id;

params[1].buffer\_type = MYSQL\_TYPE\_STRING;

params[1].buffer = title;

params[1].buffer\_length = strlen(title);

params[2].buffer\_type = MYSQL\_TYPE\_STRING;

params[2].buffer = author;

params[2].buffer\_length = strlen(author);

params[3].buffer\_type = MYSQL\_TYPE\_STRING;

params[3].buffer = date;

params[3].buffer\_length = strlen(date);

mysql\_stmt\_bind\_param(stmt, params);

mysql\_stmt\_execute(stmt); //执行与语句句柄相关的预处理

mysql\_stmt\_close(stmt);

mysql\_close(conn);

return 0;

}

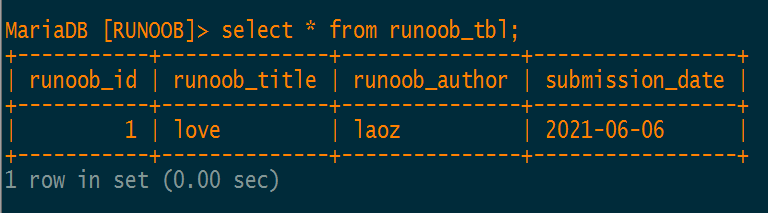
保存好之后编译



编译成功之后 我们运行，并从标准输入 输入4个数据。

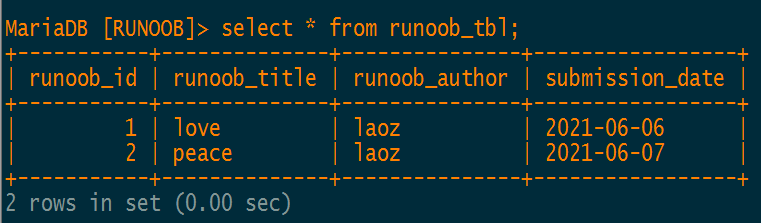


好了，这个时候 如果你已经预习了SQL语句，你可以在MYSQL 终端命令行查看这条数据是否已经成功插入了。



好了我们再执行一次吧，当然 你也可以把多次插入写在一个程序中，而不是调用多次程序。





④使用C代码来查询数据

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "mysql/mysql.h"

int main()

{

MYSQL \*conn = mysql\_init(NULL); //初始化服务器句柄

/\*登陆服务器\*/

if(!mysql\_real\_connect(conn, "localhost", "root", "myruby", "RUNOOB", 3306, NULL, 0))

{

fprintf(stderr, "mysql\_real\_connect: %s\n", mysql\_error(conn));

return -1;

}

MYSQL\_STMT \*stmt = mysql\_stmt\_init(conn); //创建MYSQL\_STMT句柄

char \*query = "select \* from runoob\_tbl;";

if(mysql\_stmt\_prepare(stmt, query, strlen(query)))

{

fprintf(stderr, "mysql\_stmt\_prepare: %s\n", mysql\_error(conn));

return -1;

}

printf("id\t title\t author\t date\t:\n ");

int id;

char title[20];

char author[20];

char date[20];

MYSQL\_BIND params[4];

memset(params, 0, sizeof(params));

params[0].buffer\_type = MYSQL\_TYPE\_LONG;

params[0].buffer = &id;

params[1].buffer\_type = MYSQL\_TYPE\_STRING;

params[1].buffer = title;

params[1].buffer\_length = 20;

params[2].buffer\_type = MYSQL\_TYPE\_STRING;

params[2].buffer = author;

params[2].buffer\_length = 20;

params[3].buffer\_type = MYSQL\_TYPE\_STRING;

params[3].buffer = date;

params[3].buffer\_length = 20;

mysql\_stmt\_bind\_result(stmt, params); //用于将结果集中的列与数据缓冲和长度缓冲关联（绑定）起来

mysql\_stmt\_execute(stmt); //执行与语句句柄相关的预处理

mysql\_stmt\_store\_result(stmt); //以便后续的mysql\_stmt\_fetch()调用能返回缓冲数据

while(mysql\_stmt\_fetch(stmt) == 0) //返回结果集中的下一行

{

printf("%d\t %s\t %s\t %s\t\n", id, title, author, date);

}

mysql\_stmt\_close(stmt);

mysql\_close(conn);

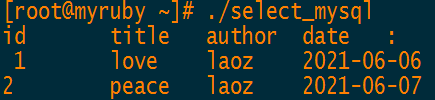
return 0;

}

编译



运行



⑤ 后续

这两个例子只是很少的一部分。同学们可以访问mysql官网学习相关函数的操作方法实践。