汐っ阳

luozhiyong131.blog.chinaunix.net

专注于嵌入式

luozhiyong131

博客访问：1477116
博文数量：941
博客积分：15795
博客等级：上将
技术积分：11974
用 户 组：普通用户
注册时间：2010-08-07 14:23

加关注 短消息

SQLite C语言接口

2011-07-22 21:56:23

分类：SQLite/嵌入式数据库

sqlite API

sqlite3 *db 是数据库连接对象,用来操作数据库

打开数据库对象函数

```
int sqlite3_open(
    const char *filename, /*数据库文件名 */
    sqlite3 **ppDb /* 创建的数据库连接对象*/
);
```

关闭数据库对象函数

```
int sqlite3_close(sqlite3 */*打开的数据库连接对象*/);
```

返回数据库错误消息

```
const char *sqlite3_errmsg(sqlite3*);
```

论坛 加好友

文章分类

全部博文 (941)
RT-Thread (2)
STM32 (25)
QT (43)
PBOC2.0 (44)
项目管理 (15)
AM1808-POS (17)
单片机 (54)
Ubuntu (8)
Android学习之旅 (97)
uCOS-II (9)
9G45-EK (51)
其他 (14)
嵌入式学习 (344)
个人日志 (5)
嵌入式实训 (69)
嵌入式自学 (76)
嵌入式学习笔记 (46)
未分配的博文 (22)

文章存档

2014年 (35)
2013年 (53)

SQLITE_OK: 表示打开成功

示例:

```
#include <stdio.h>
#include <sqlite3.h>

int main(void)
{
    /*定义一个数据库连接对象指针*/
    sqlite3 *db = NULL;
    /*初始化连接对象开辟空间*/
    int rc = sqlite3_open("sqlite.db", &db);
    if(rc != SQLITE_OK)
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"%s\n",sqlite3_errmsg(db));
        return -1;
    }
    printf("connect sucess!\n");

    sqlite3_close(db);   //闭关数据库

    return 0;
}
```

编译: gcc sqlite.c -o sqlite -I/usr/local/sqlite-3.3.7/include -

2012年 (253)
2011年 (414)
2010年 (186)


```
L/usr/local/sqlite-3.3.7/lib -lsqlite3
```

执行sql语句的函数

执行sql语句函数

```
int sqlite3_exec(
    sqlite3*, /* 打开的数据库连接对象*/
    const char *sql, /* 将要执行的sql语句*/
    int (*callback)(void*,int,char**,char**), /* 回调函数*/
    void *, /* 回调函数的第一个参数*/
    char **errmsg /* 错误的消息*/
);
```

示例:

```
#include <stdio.h>
#include <sqlite3.h>

int main(void)
{
    sqlite3 *db = NULL;
    int rc = sqlite3_open("sqlite.db", &db);
    if(rc != SQLITE_OK)
    {
        fprintf(stderr, "%s\n", sqlite3_errmsg(db));
        return -1;
```

```
    }

    char *sql1 = "create table if not exists stu(id integer primary key
default 1, name varchar(20));";
    char *sql2 = "insert into stu(name) values('aa');";
    char *errmsg;

    rc = sqlite3_exec(db, sql1, NULL, NULL, &errmsg);
    if(rc != SQLITE_OK)
    {
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
        fprintf(stderr,"错误%s\n",errmsg);
    }

    rc = sqlite3_exec(db, sql2, NULL, NULL, &errmsg);
    if(rc != SQLITE_OK)
    {
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
        fprintf(stderr,"错误%s\n",errmsg);
    }

    sqlite3_close(db);
    return 0;
}
```

查询函数

```
int sqlite3_get_table(
    sqlite3 *db,          /*数据库连接对象*/
    const char *zSql, /*将要执行的sql语句*/
    char ***pazResult, /* 查询的结果集 */
    int *pnRow,           /* 结果集的行数*/
    int *pnColumn,        /*结果集的列数*/
    char **pzErrmsg   /* 查询的错误消息*/
);
```

释放结果集函数

```
void sqlite3_free_table(char **result);
```

示例:

```
#include <stdio.h>
#include <sqlite3.h>

int main(void)
{
    /*定义一个数据库连接对象指针*/
    sqlite3 *db = NULL;
    int rc ;
    /*初始化连接对象,开辟空间*/
    rc = sqlite3_open("test.db",&db);

    if (rc != SQLITE_OK )
```

```
{
    /*获取连接对象的错误信息*/
    fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
    return -1;
}

char *sql = "select * from tbl_emp";
char **result = NULL;
int rows;
int cols;
rc = sqlite3_get_table(db,sql,&result,&rows,&cols,NULL);
if (rc != SQLITE_OK )
{
    /*获取连接对象的错误信息*/
    fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
    return -1;
};

int i, flag=0;
for(i = 0;i < (rows + 1) * cols ;i++)
{
    if((flag++) == cols)
    {
        flag=1;
        printf("\n");
    }
```

```
            printf("%12s", result[i]);
        }

        printf("\n");

        sqlite3_free_table(result);

        /*关闭数据库,释放空间*/
        sqlite3_close(db);
        return 1;
}


#include <stdio.h>
#include <sqlite3.h>

int main(void)
{
        /*定义一个数据库连接对象指针*/
        sqlite3 *db = NULL;
        int rc ;
        /*初始化连接对象,开辟空间*/
        rc = sqlite3_open("test.db",&db);

        if (rc != SQLITE_OK )
        {
```

```
    /*获取连接对象的错误信息*/
    fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
    return -1;
}

char * name="denny";
char *sql;

char buff[200];
sprintf(buff,"select * from tbl_emp where name ='%s'",name);
sql = (char *)malloc(strlen(buff));
strcpy(sql,buff);

char **result = NULL;
int rows;
int cols;
rc = sqlite3_get_table(db,sql,&result,&rows,&cols,NULL);
if (rc != SQLITE_OK )
{
    /*获取连接对象的错误信息*/
    fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
    return -1;
};
int i, flag=0;
for(i = 0;i < (rows + 1) * cols ;i++)
{
```

```
        if((flag++) == cols)
        {
            flag=1;
            printf("\n");
        }
        printf("%12s", result[i]);
    }

    sqlite3_free_table(result);

    printf("数据库连接成功!\n");
    /*关闭数据库,释放空间*/
    sqlite3_close(db);
    return 1;
}
```

回调函数

```
int (*callback)(
    void*,/*从sqlite3_exec传递来的参数*/
    int,/*结果集的列数*/
    char**, /*列的值*/
    char**/*列的名字*/
)
```

示例:

```
#include <stdio.h>
```

```c
#include <sqlite3.h>

int testcallback(void *d,int cols,char **col_values,char **col_names)
{
    int i;

    int flag  = *((int *)d);

    if (flag == 1)
    {
        for(i = 0;i< cols ;i++)
        {
            printf("%12s",col_names[i]);
        }
        printf("\n");
    }

    for(i = 0;i< cols ;i++)
    {
        printf("%12s",col_values[i]);
    }
    printf("\n");

    return 0;
}
```

```c
int main(void)
{
    /*定义一个数据库连接对象指针*/
    sqlite3 *db = NULL;
    int rc ;
    /*初始化连接对象,开辟空间*/
    rc = sqlite3_open("test.db",&db);

    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
        return -1;
    }

    printf("数据库连接成功!\n");

    /*插入数据*/
    char * sql = "select * from tbl_emp";
    char * errmsg;
    int flag = 1;
    rc = sqlite3_exec(db,sql,testcallback,&flag,&errmsg);
    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
```

```
        fprintf(stderr,"错误%s\n",errmsg);
        return -1;
    }

    /*关闭数据库,释放空间*/
    sqlite3_close(db);
    return 1;
}
```

预处理对象

```
int sqlite3_prepare(
    sqlite3 *db, /* 数据库连接对象*/
    const char *zSql, /*将要执行的sql语句*/
    int nByte, /* sql语句的长度 -1*/
    sqlite3_stmt **ppStmt, /* sqlite3_stmt对象 */
    const char **pzTail /* 指向执行的sql语句 0*/
);
int sqlite3_bind_text(sqlite3_stmt*, int, const char*, int n, void(*)
(void*));
int sqlite3_step(sqlite3_stmt*);
const unsigned char *sqlite3_column_text(sqlite3_stmt*, int iCol);
int sqlite3_finalize(sqlite3_stmt *pStmt);
```

示例:

```
#include <stdio.h>
#include <sqlite3.h>
```

```c
int main(void)
{
    /*定义一个数据库连接对象指针*/
    sqlite3 *db = NULL;
    int rc ;
    /*初始化连接对象,开辟空间*/
    rc = sqlite3_open("test.db",&db);

    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
        return -1;
    }
    char * name = "denny";
    char * sql = "select * from tbl_emp where name=? and id=?";
    sqlite3_stmt  *stmt = NULL;
    rc = sqlite3_prepare(db,sql,-1,&stmt,NULL);

    sqlite3_bind_text(stmt,1,name,strlen(name),NULL);

    sqlite3_bind_int(stmt,2,2);

    sqlite3_step(stmt);
    printf("id=%d\n",sqlite3_column_int(stmt,0));
```

```
    printf("name=%s\n",sqlite3_column_text(stmt,1));


    sqlite3_finalize(stmt);

    printf("数据库连接成功!\n");
    /*关闭数据库,释放空间*/
    sqlite3_close(db);
    return 1;
}

#include <stdio.h>
#include <sqlite3.h>

int main(void)
{
    /*定义一个数据库连接对象指针*/
    sqlite3 *db = NULL;
    int rc ;
    /*初始化连接对象,开辟空间*/
    rc = sqlite3_open("test.db",&db);

    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
```

```
        return -1;
    }

    char * sql = "select * from tbl_emp";
    sqlite3_stmt  *stmt = NULL;
    rc = sqlite3_prepare(db,sql,-1,&stmt,NULL);
    sqlite3_step(stmt);
    printf("id=%d\n",sqlite3_column_int(stmt,0));
    printf("name=%s\n",sqlite3_column_text(stmt,1));

    sqlite3_step(stmt);
    printf("id=%d\n",sqlite3_column_int(stmt,0));
    printf("name=%s\n",sqlite3_column_text(stmt,1));

    sqlite3_finalize(stmt);

    printf("数据库连接成功!\n");
    /*关闭数据库,释放空间*/
    sqlite3_close(db);
    return 1;
}
```

预处理对象重复使用

```
#include <stdio.h>
#include <sqlite3.h>
```

```c
int main(void)
{
    /*定义一个数据库连接对象指针*/
    sqlite3 *db = NULL;
    int rc ;
    /*初始化连接对象,开辟空间*/
    rc = sqlite3_open("test.db",&db);

    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
        return -1;
    }
    char * name = "annie";
    char * sql = "select * from tbl_emp where name=?";
    sqlite3_stmt  *stmt = NULL;
    rc = sqlite3_prepare(db,sql,-1,&stmt,NULL);

    sqlite3_bind_text(stmt,1,name,strlen(name),NULL);
    sqlite3_step(stmt);
    printf("id=%d\n",sqlite3_column_int(stmt,0));
    printf("name=%s\n",sqlite3_column_text(stmt,1));
    printf("-------\n");
    sqlite3_reset(stmt);
    sqlite3_bind_text(stmt,1,"denny",strlen("denny"),NULL);
```

```
    sqlite3_step(stmt);
    printf("id=%d\n",sqlite3_column_int(stmt,0));
    printf("name=%s\n",sqlite3_column_text(stmt,1));


    sqlite3_finalize(stmt);

    printf("数据库连接成功!\n");
    /*关闭数据库,释放空间*/
    sqlite3_close(db);
    return 1;
}
```

事务

事务是数据库最小处理单元

事务的四个特性:

1.原子性

2.隔离性

3.一直性

4.持久性

sqlite如何使用事务

1.begin trasaction

2.rollback --回滚（取消）

3.commit --提交

```
4.sqlite3_exec(db,"begin trasaction",NULL,NULL,NULL)
5.sqlite3_exec(db,"commit",NULL,NULL,NULL)
6.sqlite3_exec(db,"rollback",NULL,NULL,NULL)


#include <stdio.h>
#include <sqlite3.h>

int main(void)
{
    /*定义一个数据库连接对象指针*/
    sqlite3 *db = NULL;
    int rc ;
    /*初始化连接对象,开辟空间*/
    rc = sqlite3_open("test.db",&db);

    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
        return -1;
    }

    printf("数据库连接成功!\n");

    /*插入数据*/
```

```
    char * sql1 = "insert into tbl_emp(id,name,age,birthday)
values(5,'denny',32,date('1977-06-28'))";
    char * sql2 = "insert into tbl_emp(id,name,age,birthday)
values(5,'denny',32,date('1977-06-28'))";
    char * errmsg;
    rc = sqlite3_exec(db,"begin transaction",NULL,NULL,&errmsg);
    rc = sqlite3_exec(db,sql1,NULL,NULL,&errmsg);
    rc = sqlite3_exec(db,sql2,NULL,NULL,&errmsg);
    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
        fprintf(stderr,"错误%s\n",errmsg);
        sqlite3_exec(db,"rollback transaction",NULL,NULL,&errmsg);
        return -1;
    }
    sqlite3_exec(db,"commit  transaction",NULL,NULL,&errmsg);

    /*关闭数据库,释放空间*/
    sqlite3_close(db);
    return 1;
}
```

其它:

```
#include <stdio.h>
```

```c
#include <sqlite3.h>
#include <fcntl.h>
int main(void)
{
    /*定义一个数据库连接对象指针*/
    sqlite3 *db = NULL;
    int rc ;
    /*初始化连接对象,开辟空间*/
    rc = sqlite3_open("test.db",&db);

    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
        return -1;
    }

    printf("数据库连接成功!\n");

    /*插入数据*/
    char * sql = "insert into tbl_emp(id,photo) values(7,?)";

    sqlite3_stmt * stmt = NULL;
    rc = sqlite3_prepare(db,sql,-1,&stmt,NULL);

    /*读取一个文件*/
```

```
    int fd = open("logo.gif",O_RDONLY);
    int buff[1024*8];
    int len = read(fd,buff,sizeof(buff));
    sqlite3_bind_blob(stmt,1,buff,len,NULL);
    sqlite3_step(stmt);
    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));

        return -1;
    }

    /*关闭数据库,释放空间*/
    sqlite3_close(db);
    return 1;
}



#include <stdio.h>
#include <sqlite3.h>
#include <fcntl.h>
int main(void)
{
    /*定义一个数据库连接对象指针*/
```

```
sqlite3 *db = NULL;
int rc ;
/*初始化连接对象,开辟空间*/
rc = sqlite3_open("test.db",&db);

if (rc != SQLITE_OK )
{
    /*获取连接对象的错误信息*/
    fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));
    return -1;
}

printf("数据库连接成功!\n");

/*插入数据*/
char * sql = "select photo from tbl_emp where id =7";

sqlite3_stmt * stmt = NULL;
rc = sqlite3_prepare(db,sql,-1,&stmt,NULL);
sqlite3_step(stmt);
const void *pimage = sqlite3_column_blob(stmt,0);
int size = sqlite3_column_bytes(stmt,0);

/*读取一个文件*/
int fd = open("logo1.gif",O_WRONLY|O_CREAT,0777);
```

```
    write(fd,pimage,size);

    if (rc != SQLITE_OK )
    {
        /*获取连接对象的错误信息*/
        fprintf(stderr,"错误%s\n",sqlite3_errmsg(db));

        return -1;
    }

    /*关闭数据库,释放空间*/
    sqlite3_close(db);
    return 1;
}
```

shift+ctrl+"?"--注释你选定的语句
shift+ctrl+f--自动排版

上一篇：SQLite 快速入门
下一篇：SQLite SQL语法详解

0

相关热门文章

轻量级web server Tornado代码...
SQLITE3 使用总结
kernel 报错l701.exel[16922]:...
caller和callee的区别
SQLite从Excel文件中导入数据...
C语言 如何在一个整型左边补0...
C语言实现字符串变量的trim操...
SQLite SQL语法详解
python无法爬取阿里巴巴的数据...
OC 点语法的属性
sqlite3加密
linux-2.6.28 和linux-2.6.32....
移民美国益处多 美国移民日渐...
J-LINK V8(V7)固件烧录指导...
linux su - username -c 命...

给主人留下些什么吧！~~

评论热议

请登录后评论。

登录 注册