# Linux 下用 freetds 执行 SqlServer 的 sql 语句和存储过程

Windows 下访问 Sqlserver 很方便，特别是用 ADO，即便是用 C++写代码，也没怎么感觉麻烦,如果是用 C#的话，写起来估计更是飞一般的感觉，可现在我要处理的问题是在 Linux 下访问 SqlServer，执行 sql 语句和存储过程……

好，不废话了，下面开工。

一、包含头文件

```
#include <sybfront.h> //freetds
#include <sybdb.h> //freetds
```

二、执行 sql 语句或存储过程

1、查询类

1.1 核心代码：

```
bool queryCmd(DBPROCESS *dbprocess,const char* strSql)
{
    dbcmd(dbprocess,strSql);
    if(dbsqlexec(dbprocess) == FAIL)
    {
        printf("Query error.\n");
        return false;
    }
    DBINT result_code;
    char infArr[MaxColumnNums][MaxColumnSize];
    int retCode = 1;
    while ((result_code = dbresults(dbprocess)) != NO_MORE_RESULTS)
    {
        if (result_code == SUCCEED)
        {
            int i=1;
            int sz = 0;
            while(true)
            {
                //retCode = dbbind(dbprocess,i++, CHARBIND, (DBCHAR)0,
(BYTE*)infArr[i]);
                retCode = dbbind(dbprocess,i, CHARBIND, (DBINT)0,
(BYTE*)infArr[i]);
                if(retCode != 1) break;
                i++;
            }
            sz = i;
            while (dbnextrow(dbprocess) != NO_MORE_ROWS)
            {
                for(i=1; i<=sz; i++)
                {
                    //printf("%s ",infArr[i]);
                    cout<<infArr[i]<<" ";//<<endl;
                    memset(infArr[i],0,sizeof(infArr[i]));
                }
                cout<<endl;
            }
        }
    }
    return true;
}
```

1.2 直接执行 Sql 语句

```
queryCmd(dbprocess, "select * from table");
```

1.3 不带参数的存储过程

创建存储过程如下：

```
create proc testPro
as
        select * from stu;
go
```

调用如下：

```
queryCmd(dbprocess, "exec testPro");
```

1.4 带参数的存储过程

创建存储过程如下：

```
create proc getPro1(@num int)
as
        select * from stu where StuID = @num;
go
```

调用如下：

```
queryCmd(dbprocess, "exec getPro1 1003");
```

2、更新类

2.1 核心代码：

```
bool updateCmd(DBPROCESS *dbprocess,const char* strSql)
{
    dbcmd(dbprocess,strSql);
    if(dbsqlexec(dbprocess) == FAIL)
    {
        printf("error : update fail\n");
        return false;
    }
    return true;
}
```

2.2 直接执行 sql 语句

```
updateCmd(dbprocess,"insert into stu(StuID, Name, Age) values(888,'Mike',24)");
```

2.3 不带参数的存储过程

创建存储过程如下：

```
create proc delPro1
as
        delete from stu where StuID=888
go
```

调用如下：

```
updateCmd(dbprocess,"exec delPro1");
```

2.4 带参数的存储过程

创建存储过程如下：

```
create proc delPro2(@num int)
as
        delete from stu where StuID=@num
go
```

调用如下：

```
updateCmd(dbprocess,"exec delPro2 888");
```

三、编译选项
比如源文件为 test2.cpp，freetds 的安装路径为 usr/local/freetds ，则如下编译：
g++ -g test2.cpp -o test2 -L/usr/local/freetds/lib -lsybdb -I/usr/local/freetds/include

PS：
环境 freetds 0.91 + Sqlserver 2008

附完整示例代码：

```
/*
	File		: freetdsTest.cpp
	Author		: Mike
	E-Mail		: Mike_Zhang@live.com
*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include <sybfront.h> //freetds
#include <sybdb.h> //freetds

#include <string>
#include <vector>
#include <iostream>

using namespace std;

#define MaxColumnNums 255
#define MaxColumnSize 1024

bool queryCmd(DBPROCESS *dbprocess,const char* strSql)
{
	dbcmd(dbprocess,strSql);
	if(dbsqlexec(dbprocess) == FAIL)
	{
		printf("Query error.\n");
		return false;
	}
	DBINT result_code;
	char infArr[MaxColumnNums][MaxColumnSize];
	int retCode = 1;
	while ((result_code = dbresults(dbprocess)) != NO_MORE_RESULTS)
	{
		if (result_code == SUCCEED){
			int i=1;
			int sz = 0;
			while(true)
			{
				//retCode = dbbind(dbprocess,i++, CHARBIND,
(DBCHAR)0, (BYTE*)infArr[i]);
				retCode = dbbind(dbprocess,i, CHARBIND, (DBINT)0,
(BYTE*)infArr[i]);
				if(retCode != 1) break;
				i++;
			}
			sz = i;
			while (dbnextrow(dbprocess) != NO_MORE_ROWS){
```

```
                        for(i=1;i<=sz;i++)
                        {
                                //printf("%s ",infArr[i]);
                                cout<<infArr[i]<<" ";//<<endl;
                                memset(infArr[i],0,sizeof(infArr[i]));
                        }
                        cout<<endl;
                }
        }
    }
    return true;
}

bool updateCmd(DBPROCESS *dbprocess,const char* strSql)
{
    dbcmd(dbprocess,strSql);
    if(dbsqlexec(dbprocess) == FAIL)
    {
        printf("error : update fail\n");
        return false;
    }
    return true;
}

int queryIt(DBPROCESS *dbprocess)
{
    queryCmd(dbprocess, "select * from cdr");
    queryCmd(dbprocess, "exec test1Proc");
    queryCmd(dbprocess, "exec test2Proc");
    queryCmd(dbprocess, "exec getPro1 1003");
}

int insertIt(DBPROCESS *dbprocess)
{
    return updateCmd(dbprocess,"insert into stu(StuID, Name, Age)
values(888,'James',28)");
}

int deleteIt(DBPROCESS *dbprocess)
{
    //return updateCmd(dbprocess,"delete from stu where StuID=888");
    //return updateCmd(dbprocess,"exec delPro1");
    return updateCmd(dbprocess,"exec delPro2 888");
}

int main(void)
{
    char szUsername[32] = "sa";
    char szPassword[32] = "123456";
    char szDBName[32] = "testDB";
    char szServer[32] = "192.168.18.113:1433";

    dbinit();

    LOGINREC *loginrec = dblogin();
    DBSETLUSER(loginrec, szUsername);
```

```
	DBSETLPWD(loginrec, szPassword);
	DBPROCESS *dbprocess = dbopen(loginrec, szServer);
	if(FAIL == dbprocess)
	{
		printf("Conect to MS SQL SERVER fail, exit!\n");
		return -1;
	}
	printf("Connect to MS SQL SERVER success!\n");

	if(FAIL == dbuse(dbprocess, szDBName))
		printf("Open database failed!\n");
	else
		printf("Open database success!\n");

	printf("Query\n");
	queryIt(dbprocess);

	printf("Insert : %d\n",insertIt(dbprocess));

	printf("Delete : %d\n",deleteIt(dbprocess));

	dbclose(dbprocess);

	return 0;
}
```