[多线程](http://www.csdn.net/tag/%e5%a4%9a%e7%ba%bf%e7%a8%8b" \t "_blank)[qt](http://www.csdn.net/tag/qt)[signal](http://www.csdn.net/tag/signal)[class](http://www.csdn.net/tag/class)[object](http://www.csdn.net/tag/object)[thread](http://www.csdn.net/tag/thread)

POINT 1:QThread类的实例与普通类的实例没什么不同,只是运行着的run()函数会不同

例1:

class MThread :public QThread   
{   
public:   
    MThread();   
    ~MThread();   
    void run();   
    void foo();   
    ...   
       
};

class MDialog :public QDialog   
{   
    ...   
    MThread \*mythread;   
};   
MDialog::MDialog()   
{   
    mythread = new MThread;   
    ...    
}  
需要注意的是,在QT中,QThread对象的实例mythread是属于创建它的线程(线程A,即MDialog所在的线程)的,mythread的所有程序代码与数据都放在与MDialog相同的空间中.这时的mythread,就像任何普通的自己定义的类的实例一样.但是在调用mythread->start()之后,mythread的run()函数中的代码会在新的线程(线程B)中执行.在run()函数中声明的变量\实例化的对象,都属于线程B.但是mythread的所有代码,都还在存储在线程A中,只是run()函数的"执行"是在线程B中.

在MDialog中,使用

mythread->foo();  
foo()是在线程A中执行的.

在MDialog中使用

connect(this, SIGNAL(sigDialogSignal()), mythread, SLOT(slotThreadSlot()));  
当emit sigDialogSignal()时,是会在MDialog所在的线程A中执行的.因为mythread与MDialog同属于一个线程, 这时thread可以看做一个普通类的实例.另外,因为connect函数的连接方式默认是自动连接,而对同属于一个纯种的两个对象,自动连接会使用直接连接,即slot在发出signal的线程中立即执行.

例2:

#include "mthread.h" #include <QDebug> MThread::MThread(QObject \*parent)   
    : QThread(parent)   
{   
    myTimer.start(1);   
    connect(&myTimer, SIGNAL(timeout()), this, SLOT(slotPrint()));   
}   
MThread::~MThread()   
{   
}   
void MThread::run()   
{      
    for (int i = 0; i < 100; ++i) {   
        for (int j = 0 ; j < 10000; ++j) {   
            qDebug()<<"---------"<<i;   
        }   
    }   
    exec();   
}   
void MThread::slotPrint()   
{   
    qDebug()<<"==============================";   
}  
运行后出现:

...   
...   
---------9   
==============================================================   
---------9   
...   
...  
不能误以为:在一个QThread类的派生类中，run()函数中的语句在运行时，可能被本线程定时器超时slot中断. (错误)

事实上,slotPrint()在创建MThread的实例的线程中执行.

POINT 2:线程B中的对象要想接收线程A中的对象发来的signal, 必须进入exec(), 如在exec()前有死循环, 没有进入exec(), 则线程B中的对象不会收到signal.

void MThread::run()   
{   
    while(1) {   
        dosomething();  //此循环永不退出     }   
    exec();             //如果此事件循环不能进入,刚此线程不会收到任何signal }  
POINT 3:线程A中的指针可指向线程B中创建的对象实例,  这个实例属于线程B. 指针仅仅是一个地址, 而对象实例的变量/代码等都属于线程B.

例1:

class MThread : public QThread   
{   
    Q\_OBJECT   
public:   
    MThread(QObject \*parent = 0);   
    ~MThread();   
    void run();   
    MPrint \*mprint;   
};   
void MThread::run()   
{   
    mprint = new MPrint;   
    exec();   
}   
//如此声明,mprint所指向的对象属于另一个线程.例2:

class MThread : public QThread   
{   
    Q\_OBJECT   
public:   
    MThread(QObject \*parent = 0);   
    ~MThread();   
    void run();   
    MPrint \*mprint;   
private:   
    QTimer \*myT  
文章出处：DIY部落(<http://www.diybl.com/course/3_program/c/c_js/20090303/157373.html>)

imer;   
private slots:   
    void slotPrint();      
    void testFoo();   
};   
void MThread::run()   
{   
    myTimer = new QTimer;   
    mprint = new MPrint;   
    myTimer->setInterval(100);   
    connect(myTimer, SIGNAL(timeout()), this, SLOT(testFoo()), Qt::DirectConnection);   
    QTimer::singleShot(0, myTimer,SLOT(start()));   
    exec();   
}  
以上这样写run(),myTimer在run()中new,即myTimer这个指针属于旧线程,但myTimer所指向的QTimer实例的实体在新的线程中,testFoo()会在新线程中执行.

例3:

void MThread::run()   
{   
    QTimer myTimer;   
    mprint = new MPrint;   
    myTimer.setInterval(100);   
    connect(&myTimer, SIGNAL(timeout()), this, SLOT(testFoo()), Qt::DirectConnection);   
    QTimer::singleShot(0, &myTimer,SLOT(start()));   
    //testFoo();     exec();   
}  
以上这样写run(),myTimer在run()中声明,即myTimer属于新的线程,testFoo()也会在新线程中执行.

例4:

class MThread : public QThread   
{   
    Q\_OBJECT   
public:   
    MThread(QObject \*parent = 0);   
    ~MThread();   
    void run();   
    MPrint \*mprint;   
private:   
    QTimer myTimer;   
private slots:   
    void slotPrint();      
    void testFoo();   
};   
void MThread::run()   
{   
    mprint = new MPrint;   
    myTimer.setInterval(100);   
    connect(&myTimer, SIGNAL(timeout()), this, SLOT(testFoo()));   
    QTimer::singleShot(0, &myTimer,SLOT(start()));   
    //testFoo();     exec();   
}  
以上这样写run(),testFoo()会在创建myTimer的老线程中执行.因为可以看到,mytimer和this(即mythread),都是在同一个线程中,只是在另一个线程中(run()),做了connect操作.

要注意的是,在线程B中启动线程A中的一个定时器,不能使用myTimer.start(),这样启动不了定时器.而应使用signal来触发start()这个slot.

POINT 5:slot不会中断同线程中的slot.

例1:

#include "mthread.h" #include <QDebug> MThread::MThread(QObject \*parent)   
    : QThread(parent)   
{   
    myTimer.start(1);   
    connect(&myTimer, SIGNAL(timeout()), this, SLOT(slotPrint()));   
}   
MThread::~MThread()   
{   
}   
void MThread::run()   
{   
    exec();   
}   
void MThread::slotPrint()   
{   
    qDebug()<<"===========================";   
    for (int i = 0; i < 100; ++i) {   
        for (int j = 0 ; j < 10000; ++j) {   
            qDebug()<<"---------"<<i;   
        }   
    }   
}   
slotPrint()函数运行完之后才会退出,说明slot不会中断slot,一个slot在执行完之后才会执行下一个slot.

注意:slotPrint()在创建MThread实例的线程中执行.而不是使用thread->start()创建出的那个线程.

例2:

#include "mthread.h" #include <QDebug> MThread::MThread(QObject \*parent)   
    : QThread(parent)   
{   
    myTimer.start(1);   
    connect(&myTimer, SIGNAL(timeout()), this, SLOT(slotPrint()));   
}   
MThread::~MThread()   
{

文章出处：DIY部落(<http://www.diybl.com/course/3_program/c/c_js/20090303/157373_2.html>)

}   
void MThread::run()   
{   
    testFoo();   
    exec();   
}   
void MThread::slotPrint()   
{   
    qDebug()<<"=======================";   
}   
void MThread::testFoo()   
{   
    for (int i = 0; i < 100; ++i) {   
        for (int j = 0 ; j < 10000; ++j) {   
            qDebug()<<"---------"<<i;   
        }   
    }   
}   
以上代码中,slotPrint()与testFoo()会在两个不同的线程中执行.