信号与槽用于对象间的通讯。产生的原因：希望任何一类的对象可以与其它类进行通讯。

当一个特定事件发生的时候，一个信号被发射，Qt的窗口部件有很多预定义的信号，也可以通过继承来加入自己的信号。槽就是一个可以被调用处理特定信号的函数，Qt的窗口部件有很多预定义的槽，也可以继承来加入自己的槽。

信号和槽：一个信号的签名（即形参）必须与它的接收槽的签名相匹配。（实际上一个槽的签名可以比它接收的信号的签名少，因为它可以忽略额外的签名）。因为签名是一致的，编译器就可以帮助我们检测类型不匹配。

所有从QObject或其子类（例如QWidget）派生的类都能够包含信号和槽。当对象改变其状态时，信号就由该对象发射出去，这就是对象所要做的全部事情，它并不知道谁在接收这个信号。槽用于接收信号，它们是普通的对象成员函数。

所有包含信号和槽的类都必须在类的顶部声明Q\_OBJECT，它们必须直接或间接派生与QObject类。

可以将很多信号与单个的槽进行连接，也可以将单个信号与很多的槽进行连接，还可以将一个信号与另一个信号连接，这时无论第一个信号什么时间发射，系统都将立即发射第二个信号。

只有定义过某个信号的类及其派生类才能够发射这个信号。当某一个信号被发射时，与其相关联的槽就被立刻执行，就象一个正常的函数调用。

如果某个信号与多个槽相连接，那么当这个信号发射时，这些槽将会一个接一个地执行，但它们执行的顺序将会是随机的。

信号的声明是在头文件中进行的，Qt的关键字signals指出进入信号声明区。例

signals:

void mySignal();

信号没有函数体定义，而且没有返回值。信号由moc自动产生，不应该在.cpp文件中产生。

槽是普通的C++成员函数，可以被正常调用，唯一的特殊性就是很多信号可以与其相关联。当与其相关联的信号发射时，这个槽会被调用。

槽可以带有参数，但其参数不能有缺省值。

槽也有存取权限，其权限决定了谁能够与其相关联。

public slots：任何对象都可将信号与之相连接。

protected slots：当前类及其子类可以将信号与之相连接。

private slots：只有类自己可以将信号与之相连接。

槽可以声明为虚函数。

通过调用QObject对象的connect函数来将某个对象的信号与另外一个对象的槽函数相关联，这样当发射者发射信号时，接收者的槽函数将被调用。

connect函数原型：

bool QObject::connect(const QObject \* sender, const char \* signal, const QObject \* receiver, const char \* member ) [static]

把从sender对象发送的signal和receiver对象中的member槽函数连接起来，连接成功返回真，否则返回假。

指定signal信号时必须使用宏SIGNAL，指定member槽函数时，必须使用宏SLOT()。

当信号与槽没有必要继续保持关联时，可以使用disconnect函数来断开连接。

disconnect函数原型：

bool QObject::disconnect(const QObject \* sender, const char \* signal, const QObject \* receiver, const char \* member ) [static]

这个函数断开发射者信号与接收者中的槽的关联。

信号与槽函数的声明一般都位于头文件中。

编写头文件时要注意添加：

#ifndef UI\_THIRD\_H

#define UI\_THIRD\_H

#endif

槽函数为虚函数：

注意：程序最好分为.h文件和.cpp文件，否则有可能会爆出链接错误：

error LNK2001: 无法解析的外部符号"public: virtual struct QMetaObject const \* \_\_thiscall A::metaObject(void)const " (?metaObject@A@@UBEPBUQMetaObject@@XZ)

#ifndef A\_H

#define A\_H

#include <QObject>

class A : public QObject

{

Q\_OBJECT

public:

A(QObject\* parent = NULL);

signals:

void signalTest();

public:

void SendSignal();

};

#endif

#include "a.h"

A::A(QObject\* parent /\* = NULL \*/)

: QObject(parent)

{

}

void A::SendSignal()

{

emit signalTest();

}

#ifndef B\_H

#define B\_H

#include <QObject>

#include <QtDebug>

class B : public QObject

{

Q\_OBJECT

public slots:

virtual void SlotTest();

};

#endif

#include "b.h"

void B::SlotTest()

{

qDebug() << "B receive signal";

}

#ifndef C\_H

#define C\_H

#include "b.h"

class C : public B

{

Q\_OBJECT

public slots:

void SlotTest();

};

#endif

#include "c.h"

void C::SlotTest()

{

qDebug() << "C receive signal";

}

#include <QApplication>

#include "a.h"

#include "b.h"

#include "c.h"

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

{

QApplication app(argc,argv);

A a;

B\* b = new C;

QObject::connect(&a, SIGNAL(signalTest()), b, SLOT(SlotTest()));

// 发射信号后，调用的是类C的SlotTest()槽函数

a.SendSignal();

return app.exec();

}

多线程下的信号与槽

Qt信号与槽的连接方式有以下几种：

QueuedConnection：向receiver所在线程的消息循环发送事件，此事件得到处理时会调用slot，像Win32的::PostMessage。

BlockingQueuedConnection：处理方式和QueuedConnection相同，但发送信号的线程会等待信号处理结束再继续，像Win32的::SendMessage。

DirectConnection：在当前线程直接调用receiver的slot，这种类型无法支持跨线程的通信。

AutoConnection：当前线程与receiver线程相同时，直接调用slot，否则同QueuedConnection类型。

如果发射者和接收者不在同一个线程中，则接收者所处的线程必须启动事件循环，否则接收不到信号

connect()函数是线程安全的。