昵称：樱桃小锤子
园龄：3年5个月
粉丝：23
关注：2
+加关注

< 2011年8月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

Android(6)
C++(5)
Qt(5)
科普(3)
NDK(2)
Python(2)

随笔档案(11)

2013年12月 (1)
2013年4月 (2)
2013年3月 (1)
2012年12月 (2)
2011年10月 (1)
2011年8月 (4)

最新评论

Qt那点事儿（二）

第二回 拒签，不只天朝有

上回说到，柯南君展示了Qt无耻的一面后，众道友心灰意冷。修仙的修仙，搞基的搞基。原本热闹的Qt道场，一下子冷清了下来。只剩哥一个人坐在联想大厦的塑像前作冥想状，“世界如果失去了Qt会怎样”？会变成知书识仪，天下大同的理想世界吗？中国五千年的24部血泪史昭示了”This is just a dream“。一个无耻倒下了，一个更无耻的又站起来。这时大屏幕上的图片映入贫道的眼帘，

霎那间贫道的内心被震撼了，终于领悟了修仙的真谛，那就是“誓将揭露Qt无耻的事业进行到底”。此时的哥又斗志昂扬，走在了修仙的大道上。

各位道友请看此例，

```
void NotifyOtherThread(DWORD dwOtherThreadID)
{
    CString str = "i am here";
    ::PostThreadMessage(dwOtherThreadID,
        WS_NOTIFY,
        &str,
        NULL);

}
```

我们知道在Windows的世界里，用Post跨线程发送消息是异步的，如果按照上面的实例，把local 类（结构）变量作为参数发送，那么接受方得到的参数内容会是混乱的，从而引起程序的崩溃。因为他一出函数体就被释放掉了。所以，我们从小就被告知，local类变量不能作为Post的参数。

那么在Qt中还是这样吗？

```
1 class MyTestA : public QObject
```

1. Re:Qt那点事儿（一）

整个求知的过程值得我们学习

--Tony.Works


```
{
    Q_OBJECT
public:
    void emitSignal()
    {
       signalMyTestA("I am here.");
    }

public slots:
    void slotMyTestA()
    {
        qDebug()<<"slotMyTestA is called.";
    }
signals:
    void signalMyTestA(QString str);
};

class MyTestB : public QObject
{
    Q_OBJECT
public slots:
    void slotMyTestB(QString str)
    {
        qDebug()<<"string is "<<str;
    }
signals:
    void signalMyTestB();
};

extern MyTestB *g_pMyTestB;
extern MyTestA *g_pMyTestA;
class MyTestC : public QThread
{
    Q_OBJECT
public:

    MyTestC():QThread(NULL)
    {
    }

    void run()
    {
        MyTestA a;
        QObject::connect(&a,SIGNAL(signalMyTestA(QString)),g_pMyTestB,SLOT(slotMyTestB(QString)));
        a.emitSignal();
        exec();
    }

public slots:
    void slotMyTestC()
```

```
    {
        qDebug()<<"slotMyTestC is called.";
    }
signals:
    void signalMyTestC();


};

//////////////////////

MyTestB *g_pMyTestB = NULL;
MyTestA *g_pMyTestA = NULL;
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);


    MyTestB b;
    g_pMyTestB = &b;

    MyTestC c;
    c.start();



    return app.exec();
}
```

只见输出窗口打出了：string is "I am here." 居然没有发生对象混乱，程序安然无恙，Local QString参数完美地被发送到了另外的线程。众道友的失落的心又重新热了起来，内心对Qt又充满了无限的期望。但俗话说得好，一朝被蛇咬，十年怕井绳。一些细心的道友，又把QString换成了QByteArray，QDate，QPixmap等常用的Qt类，无一错乱。十分，十分，只见铺天盖地的写有十的积分牌举了起来。此时语言已经是多余的了。终于不用再为Local变量烦恼了，跨线程不能使用Local变量这一无耻的行规终于被扫入了历史的垃圾箱。从此Coder道友们站起来了.....。在一片赞扬声中，贫道则陷入了深深的思索中，事有反常即为妖。

突然思绪一闪，柯南君的音容相貌又浮现在了眼前。此时贫道把代码稍微改写了下

```
struct sOwn
{
   QString m_str;
};

class MyTestA : public QObject
{
    Q_OBJECT
public:
    void emitSignal()
    {
        sOwn s;
        s.m_str = "i am here";
        signalMyTestA(s);
    }
```

```
16
17  public slots:
18      void slotMyTestA()
19      {
20          qDebug()<<"slotMyTestA is called.";
21      }
22  signals:
23      void signalMyTestA(sOwn sItem);
24  };
25
26  class MyTestB : public QObject
27  {
28      Q_OBJECT
29  public slots:
30      void slotMyTestB(sOwn sItem)
31      {
32          qDebug()<<"string is "<<sItem.m_str;
33      }
34  signals:
35      void signalMyTestB();
36  };
37
38  extern MyTestB *g_pMyTestB;
39  extern MyTestA *g_pMyTestA;
40  class MyTestC : public QThread
41  {
42      Q_OBJECT
43  public:
44
45      MyTestC():QThread(NULL)
46      {
47      }
48
49      void run()
50      {
51          MyTestA a;
52          QObject::connect(&a,SIGNAL(signalMyTestA(sOwn)),g_pMyTestB,SLOT(slotMyTestB(sOwn)));
53          a.emitSignal();
54          exec();
55      }
56
57  public slots:
58      void slotMyTestC()
59      {
60          qDebug()<<"slotMyTestC is called.";
61      }
62  signals:
63      void signalMyTestC();
64
65
```

```
};

//////////////////////////////

MyTestB *g_pMyTestB = NULL;
MyTestA *g_pMyTestA = NULL;
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);


    MyTestB b;
    g_pMyTestB = &b;

    MyTestC c;
    c.start();



    return app.exec();
}
```

一道闪电从空中划过，当代码执行到"QObject::connect(&a,SIGNAL(signalMyTestA(sOwn)),g_pMyTestB,SLOT(slotMyTestB(sOwn)));"这一句时，”QObject::connect: Cannot queue arguments of type 'sOwn' “刺眼地跳了出来。这句话通俗地说就是，你被发好人卡了，你被拒签了..., 总之就是因为你的对象不是Q家人，你就进不了Q家门，自然就connect不了slot。

这一刻Qt你又赢了，你又一次玩弄了众道友的感情，你继承了中国足球的光荣的传统。各届国家队在这一刻灵魂附体！在这一刻你不是一个人在战斗！你就像中国足球那样，在人绝望的时候给人希望，在希望的时候给人以绝望。

此时众道友仰天长叹，大声质问道，Qt你来自希望国，秉承着“All object is created equal”的信念（Qt内牛满面，哥本挪威人，奈何成花旗），却行歧视之勾当，意欲何为？更有欲到花旗传道的道友，挥舞着印有No Pass的护照，叹道，“人被拒签，难道自己写的程序也要被拒签，朗朗乾坤，公理何在，天理何存”。说完就要自碎金丹，化作散仙，不再为这个充满着无耻之徒的尘世所困扰。就在这时，一道红光飞过，仔细一看是一条鲜红的红领巾,上面写着几个屎黄色的希望文：

"qRegisterMetaType"

落款人：红领巾。

贫道急忙把此函数放到代码中，

```
struct sOwn
{
   QString m_str;
};

class MyTestA : public QObject
{
    Q_OBJECT
public:
    void emitSignal()
    {
        sOwn s;
        s.m_str = "i am here";
        signalMyTestA(s);
    }

```

```
public slots:
    void slotMyTestA()
    {
        qDebug()<<"slotMyTestA is called.";
    }
signals:
    void signalMyTestA(sOwn sItem);
};

class MyTestB : public QObject
{
    Q_OBJECT
public slots:
    void slotMyTestB(sOwn sItem)
    {
        qDebug()<<"string is "<<sItem.m_str;
    }
signals:
    void signalMyTestB();
};

extern MyTestB *g_pMyTestB;
extern MyTestA *g_pMyTestA;
class MyTestC : public QThread
{
    Q_OBJECT
public:

    MyTestC():QThread(NULL)
    {
    }

    void run()
    {
        MyTestA a;
        qRegisterMetaType<sOwn>("sOwn");
        QObject::connect(&a,SIGNAL(signalMyTestA(sOwn)),g_pMyTestB,SLOT(slotMyTestB(sOwn)));
        a.emitSignal();
        exec();
    }

public slots:
    void slotMyTestC()
    {
        qDebug()<<"slotMyTestC is called.";
    }
signals:
    void signalMyTestC();


```

```
};
////////////////////

MyTestB *g_pMyTestB = NULL;
MyTestA *g_pMyTestA = NULL;
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);



    MyTestB b;
    g_pMyTestB = &b;

    MyTestC c;
    c.start();




    return app.exec();
}
```

string is "I am here." 重回大地，众人又开始捻胡微笑，皆叹Qt的人性化。不用收入证明，不用公司邀请信，不用签证面试，只要你用了qRegisterMetaType，你就是Q家人了。

所以，当**signal**异步调用**slot**时，如果你的对象不是**Qt**所支持的，就需要调用**qRegisterMetaType**，而后就可以将**Local class**变量放到**signal**中，安全地传递给**slot**。

一切就是这么简单。Ye!!!,众道友做了个00后的标准动作，以表达自己的欢乐之情。

但是快乐的时光总是短暂的，又一幕悲剧拉开了序幕。

话说C++三朝元老，int。最近痴迷欧美音乐，给自己取了个鼎鼎响亮的艺名"雷帝嘎嘎"（LADYGAGA）。结果，

```
typedef int LADYGAGA;

struct sOwn
{
   QString m_str;
};

class MyTestA : public QObject
{
    Q_OBJECT
public:
    void emitSignal()
    {
        int i = 9;
        signalMyTestA(i);
    }

public slots:
    void slotMyTestA()
    {
        qDebug()<<"slotMyTestA is called.";
    }
```

```
signals:
    void signalMyTestA(LADYGAGA lg);
};

class MyTestB : public QObject
{
    Q_OBJECT
public slots:
    void slotMyTestB(LADYGAGA lg)
    {
        qDebug()<<"string is "<<lg;
    }
signals:
    void signalMyTestB();
};

extern MyTestB *g_pMyTestB;
extern MyTestA *g_pMyTestA;
class MyTestC : public QThread
{
    Q_OBJECT
public:

    MyTestC():QThread(NULL)
    {
    }

    void run()
    {
        MyTestA a;
        QObject::connect(&a,SIGNAL(signalMyTestA(LADYGAGA)),g_pMyTestB,SLOT(slotMyTestB(LADYGAGA)));
        a.emitSignal();
        exec();
    }

public slots:
    void slotMyTestC()
    {
        qDebug()<<"slotMyTestC is called.";
    }
signals:
    void signalMyTestC();


};
//////////////////////////////
MyTestB *g_pMyTestB = NULL;
MyTestA *g_pMyTestA = NULL;
int main(int argc, char *argv[])
{
```

```
    QApplication app(argc, argv);


    MyTestB b;
    g_pMyTestB = &b;

    MyTestC c;
    c.start();



    return app.exec();
}
```

本年度最具爆炸性新闻诞生了“QObject::connect: Cannot queue arguments of type 'LADYGAGA'，(Make sure 'LADYGAGA' is registered using qRegisterMetaType().)” 。世界震惊了，大家宁可相信“发改委宣布降低油价”，也不愿相信，编程界基本数据类型的三朝元老，久经考验的C++主义战士，int同志居然因为使用众人皆知的艺名connect slot，被拒签了。三大社（美联社，法新社，路透社）联合发文<<悲剧，三朝元老使用艺名，被拒签>>。2月30号X民日报头版发表评论员文章《老同志，倚老卖老要不得》。

从上诉评论中我们不难看出"当signal和slots是异步时，哪怕数据类型用qRegisterMetaTyoe注册过,甚至是基本数据类型，connect用的参数类型不是注册名，而是typedef或者define的别名，connect也会失败，如果要使用别名，也需要对别名进行重新注册。**Qt**只认名，不认人"。对于上面的例子，如果要使connect能正常使用，需调用qRegisterMetaType<LADYGAGA>("LADYGAGA");，虽然LADYGAGA其实就是int。

Qt总是在意想不到的时间，意想不到的地点，展示其无耻的一面。

太史公曾曰，世上无耻者，皆不过Qt也。

古人不曾欺我。

而众道友此时却没有因为Qt的再次无耻，而愤愤不平，反而露出一丝猥琐的笑容，心里不禁乐道，老东西，你也有今天。

而贫道又陷入了沉思，Qt你的底线到底在哪里？

欲知后事如何，请看下回分解。

标签: C++, Qt

樱桃小锤子
关注 - 2
粉丝 - 23
+加关注

« 上一篇：Qt那点事儿（一）
» 下一篇：Qt那点事儿（三） 论父对象与子对象的关系

posted on 2011-08-08 09:40 樱桃小锤子 阅读(2416) 评论(8) 编辑 收藏

评论：

#1楼 2011-08-08 09:56 | helloworld2
qt是什么

支持(0) 反对(0)

#2楼 2011-08-08 10:00 | gardensu
不错，各方面都考虑到了。而且语言文字诙谐，轻松。
期待下文！

支持(0) 反对(0)

#3楼 2011-08-08 10:05 | gardensu
回一楼，QT是一套GUI解决方案。起初，用来开发嵌入式Liunx的用户界面。后来被诺基亚收购，准备用来开发Meego应用程序。后来诺基亚放弃Meego，QT前程就有些迷茫了。

支持(0) 反对(0)

#4楼 2011-08-08 10:57 | 小静（Cathy）
楼主蛮风趣的

支持(0) 反对(0)

#5楼[楼主] 2011-08-08 11:24 | 樱桃小锤子
@丫头小静（Cathy）
静师太来访，贫道有失远迎

支持(0) 反对(0)

#6楼 2011-08-08 11:32 | 子曰2[未注册用户]
QT老是在升级。刚入道的一小道士求各位老道士用的是啥版本的QT,哪个版本比较稳定

#7楼[楼主] 2011-08-08 12:34 | 樱桃小锤子
@子曰2
我用的是4.7，只要是4.X的都差不多，对你自己写的代码来说没什么需要改变的。不过上次膜拜Qt大神的时候，大神告诉了我个消息，今年秋天会发布5.0.而且我在他机器里也发现了5.0的代码。

支持(0) 反对(0)

#8楼 2011-08-08 14:46 | testzhangsan
引用

樱桃小锤子：
@丫头小静（Cathy）
静师太来访，贫道有失远迎

这句话亮了！

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新**IT**新闻：
· 刷大墙/送财神/送春联 网贷平台进军农村也是拼了！
· 阿里与工商总局掐架 受伤的是美国股民
· 为啥手机厂商一窝蜂地烧Hi-Fi，做耳机？
· 你在淘宝上的买买买，能让阿里巴巴给你的信用打几分呢？
· 亚马逊将分拆AWS云计算服务
» 更多新闻...

最新知识库文章：
· 大数据架构和模式（五）——对大数据问题应用解决方案模式并选择实现它的产品
· 大数据架构和模式（四）——了解用于大数据解决方案的原子模式和复合模式
· 大数据架构和模式（三）——理解大数据解决方案的架构层
· 大数据架构和模式（二）——如何知道一个大数据解决方案是否适合您的组织
· 大数据架构和模式（一）——大数据分类和架构简介
» 更多知识库文章...