**Qt信号槽处理机制**

一 某个对象在QCoreApplication::exec之前emit信号

1, emit调用acctive,根据该对象的connectionLists会逐一触发每一个接收者的槽,如果是同一线程,则调用qt\_static\_metacall立即触发槽函数(无需等到线程调度器),如果接收者是其它线程,则queued触发,调用postEvent(QMetaCallEvent),放入其它线程的postEventList,随后调用所属线程调度器的wakeup(此时由于线程调度器内部的hwd未建立,所以没有处理),后续过程与第二deleteLater的介绍相同(换成消息QMetaCallEvent)

二 某个对象在QCoreApplication::exec之前deleteLater

1, deleteLater调用postEvent(QDeferredDeleteEvent),放入对象所属线程的postEventList中,随后调用所属线程调度器的wakeup(此时由于线程调度器内部的hwd未建立,所以没有处理)

2, 线程调度器在启动后调用wakeup,给hwd发送WM\_QT\_SENDPOSTEDEVENTS事件

3, 线程调度器循环中,遇到hwd的WM\_QT\_SENDPOSTEDEVENTS事件,取出postEventList中的事件,调用sendPostedEvents(),最终触发sendEvent(QObject \*receiver, QEvent \*event),进入QObject的event函数中,case QEvent::DeferredDelete,析构对象,emit destroyed信号

三 某个定时器在QCoreApplication::exec之前start

1, start函数参数0为QZeroTimerEvent，小于20毫秒为fastTimerId，其它为SetTimer。

1，start调用registerTimer,将定时器参数写入线程调度器的timerDict

2, 线程调度器在启动后调用createInternalHwnd,根据定时器参数选择调用三者之一,postEvent(QZeroTimerEvent),SetTimer,fastTimerId。QZeroTimerEvent是在start的参数internal为0时post，以下3，4步是处理QZeroTimerEvent过程，第5步是处理SetTimer过程，第6步为处理fastTimerId。

3, 线程调度器接着调用wakeup,给hwd发送WM\_QT\_SENDPOSTEDEVENTS事件

4, 线程调度器循环中,遇到hwd的WM\_QT\_SENDPOSTEDEVENTS事件,取出postEventList中的事件,调用sendPostedEvents(),后续处理与第二deleteLater介绍中的相同(换成消息QZeroTimerEvent)

5, 线程调度器循环中,遇到SetTimer超时消息,会调用sendTimerEvent(timerId),最终触发sendEvent(QObject \*receiver, QEvent \*event),进入QTimer基类QObject的event函数中,case QEvent::Timer,最终emit timeout信号,后续通过activate调用connectionLists中的槽

6, fastTimerId线程超时后,会跨线程调用postEvent(QTimerEvent),放入对象线程的postEventList中,随后后调用事件处理的wakeup(此时线程调度器内部的hwd如果建立,就可以立即给hwd发送WM\_QT\_SENDPOSTEDEVENTS事件)

四 QSocketNotifier对象在QCoreApplication::exec之前创建

1, new QSocketNotifier时调用registerSocketNotifier,将fd放入sn\_read中,随后调用postActivateSocketNotifiers,给hwd发送WM\_QT\_ACTIVATENOTIFIERS事件

2, 线程调度器循环中,遇到hwd的WM\_QT\_ACTIVATENOTIFIERS事件,通过调用doWsaAsyncSelect,注册WM\_QT\_SOCKETNOTIFIER事件.

3, 线程调度器循环中,遇到WM\_QT\_SOCKETNOTIFIER消息(例如接收到对方udp报文),最终触发sendEvent(QEvent::SockAct),进入QSocketNotifier的event函数中(注意QSocketNotifier的event处理只是emit activated信号,并没有槽函数来处理数据导致一直会产生该信号,所以必须继承QSocketNotifier重写event函数）

4, QSocketNotifier的event处理中首先处理QEvent::ThreadChange,接着调用了QObject::event(e),然后再处理 QEvent::SockAct和QEvent::SockClose,所以在派生QSocketNotifier时要注意基类的event.派生QSocketNotifier时只处理QEvent::SockAct和QEvent::SockClose,其它仍由QSocketNotifier::event处理

五 QTimer::singleShot可以触发另外一个对象的槽函数,这个槽函数必须是无参的.但即使第一个参数是0msec,槽函数也需要等到线程调度器循环中才能触发,因为0msec时是queued触发. QTimer::singleShot是通过调用QMetaObject::invokeMethod实现的.

六 一个QObject对象在QCoreApplication::exec之前析构时，如果它曾经有postedEvents过的消息，则在它的析构函数里就会调用QCoreApplication::removePostedEvents，用来清除掉posted到该对象隶属线程的事件队列里的以该对象为receiver方的所有post事件。

七 事件调度器共建立了3个事件,WM\_QT\_SOCKETNOTIFIER和WM\_QT\_ACTIVATENOTIFIERS用于处理第四中的SocketNotifier,WM\_QT\_SENDPOSTEDEVENTS用于处理postEventList中的事件

enum {

WM\_QT\_SOCKETNOTIFIER = WM\_USER,

WM\_QT\_SENDPOSTEDEVENTS = WM\_USER + 1,

WM\_QT\_ACTIVATENOTIFIERS = WM\_USER + 2,

SendPostedEventsWindowsTimerId = ~1u

};