Linux程序开发：QT中的多线程编程（3） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022_Linux_E7_A8_8B_E5_BA_c103_144790.htm 3、用户自定义事件在多线程编程中的应用 在 Qt 系统中，定义了很多种类的事件，如定时器事件、鼠标移动事件、键盘事件、窗口控件事件等。通常，事件都来自底层的窗口系统，Qt 的主事件循环函数从系统的事件队列中获取这些事件，并将它们转换为 QEvent，然后传给相应的 QObjects 对象。 除此之外，为了满足用户的需求，Qt 系统还提供了一个 QCustomEvent 类，用于用户自定义事件，这些自定义事件可以利用 QThread::postEvent() 或者QApplication::postEvent() 被发给各种控件或其他 QObject 实例，而 QWidget 类的子类可以通过 QWidget::customEvent() 事件处理函数方便地接收到这些自定义的事件。需要注意的是：QCustomEvent 对象在创建时都带有一个类型标识 id 以定义事件类型，为了避免与 Qt 系统定义的事件类型冲突，该 id 值应该大于枚举类型 QEvent::Type 中给出的 "User" 值。 在下面的例子中，显示了多线程编程中如何利用用户自定义事件类。UserEvent类是用户自定义的事件类，其事件标识为346798，显然不会与系统定义的事件类型冲突。 class UserEvent : public QCustomEvent //用户自定义的事件类 { public: UserEvent(QString s) : QCustomEvent(346798), sz(s) { . } QString str() const { return sz. } private: QString sz. }. UserThread类是由QThread类继承而来的子类，在该类中除了定义有关的变量和线程控制函数外，最主要的是定义线程的启动函数UserThread::run()，在该函数中创建了一个用户自定义事

件UserEvent，并利用QThread类的postEvent函数提交该事件给相应的接收对象。 class UserThread : public QThread //用户定义的线程类 { public: UserThread(QObject *r, QMutex *m, QWaitCondition *c). QObject *receiver. } void UserThread::run() //线程类启动函数，在该函数中创建了一个用户自定义事件 {UserEvent *re = new UserEvent(resultstring). QThread::postEvent(receiver, re). } UserWidget类是用户定义的用于接收自定义事件的QWidget类的子类，该类利用slotGo()函数创建了一个新的线程recv（UserThread类），当收到相应的自定义事件（即id为346798）时，利用customEvent函数对事件进行处理。 void UserWidget::slotGo() //用户定义控件的成员函数 { mutex.lock(). if (! recv) recv = new UserThread(this, amp.condition). recv->start(). mutex.unlock(). } void UserWidget::customEvent(QCustomEvent *e) //用户自定义事件处理函数 { if (e->type()==346798) { UserEvent *re = (UserEvent *) e. newstring = re->str(). } } 在这个例子中，UserWidget对象中创建了新的线程UserThread，用户可以利用这个线程实现一些周期性的处理（如接收底层发来的消息等），一旦满足特定条件就提交一个用户自定义的事件，当UserWidget对象收到该事件时，可以按需求做出相应的处理，而一般情况下，UserWidget对象可以正常地执行某些例行处理，而完全不受底层消息的影响。 