ROS消息队列清空说明

李睿豪 2018.11.02

一、问题描述

MicROS角色线程暂停，暂停期间，相应订阅者（subscriber）正常运行消息（message）订阅，在重启角色后，订阅者的回调函数（callback）首先调用处理过去的消息。根据MicROS角色要求，应对过去消息队列进行清空，实现在每次重启角色时，直接调用回调函数处理当前消息。

二、ROS Kinetic源代码说明

在本问题中，消息队列有两个，一个订阅中的消息队列（详见subscription\_queue.cpp），一个是回调函数中的消息队列（详见callback\_queue.cpp）。

ROS Kinetic在CallbackQueue类（callback\_queue.cpp）中提供了CallbackQueue::clear()函数，对上层用户可见。但分析源代码发现，CallbackQueue队列中，存放的是对订阅者消息队列操作的指针，CallbackQueue队列为n，可以通俗理解为对订阅者消息队列进行n次操作，通过调用CallbackQueue::clear()函数对CallbackQueue队列进行清空，并不能实现对订阅者消息队列的清空，重启角色时，回调函数依然会优先处理存放在订阅者消息队列中的旧消息。

在SubscriptionQueue类（subscription\_queue.cpp）中，提供了SubscriptionQueue::clear()函数，此函数对上层用户不可见。同时，根据角色概念，每一个订阅者需要对应一个回调函数队列，因此，需要通过修改CallbackQueue类（callback\_queue.cpp），通过CallbackQueue对象，同时对相应订阅者的消息队列和回调函数消息队列进行清空。

三、ROS Kinetic版本源代码修改说明

修改源代码目录：/src/ros\_comm/roscpp/

修改代码文件：/include/ros/callback\_queue\_interface.h

/include/ros/callback\_queue.h

/include/ros/subscription\_interface.h

/src/libros/callback\_queue.cpp

/src/libros/subscription\_queue.cpp

* /include/ros/callback\_queue\_interface.h修改说明：

在CallbackInterface基类中，添加虚函数。

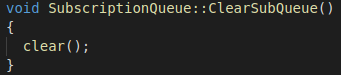
C:\Users\ADMINI~1\AppData\Local\Temp\1540954046(1).png

* /include/ros/subscription\_queue.h修改说明：

在SubscriptionQueue子类中，添加虚函数声明。

C:\Users\ADMINI~1\AppData\Local\Temp\1540954089(1).png

* /src/libros/subscription\_queue.cpp修改说明：

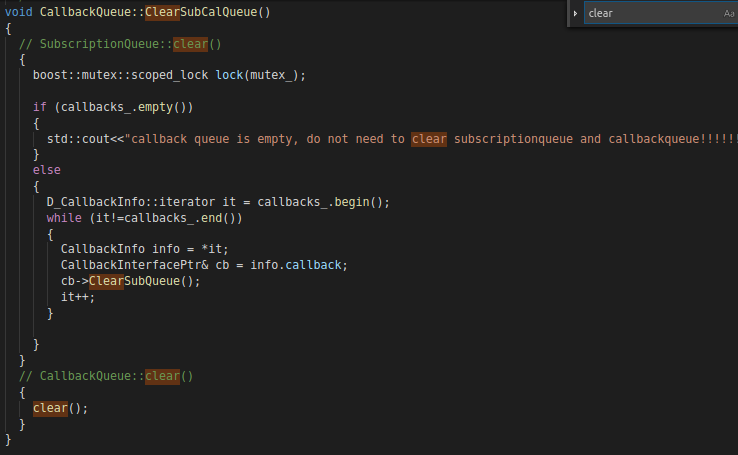
 实现对SubscriptionQueue::clear（）的访问，清空订阅消息队列。

* /include/ros/callback\_queue.h修改说明：

在CallbackQueue类中，添加用户可见的ClearSubCalQueue()函数，实现对订阅者和回调函数消息队列的清空。

C:\Users\ADMINI~1\AppData\Local\Temp\1540954361(1).png

* /src/libros/callback\_queue.cpp修改说明：

在CallbackQueue类中，添加用户可见的ClearSubCalQueue()函数，实现对订阅者和回调函数消息队列的清空。

* 源码编译说明：

ROS Kinetic的源代码修改完毕后，进入源代码所在目录（home/XXX/ros\_catkin\_ws）,只对修改的roscpp包极其依赖进行重新编译:

./src/catkin/bin/catkin\_make\_isolate --install --only-pkg-with-deps roscpp

四、测试说明

节点1（node 1，命名talker）通过发布者（publisher）以1Hz的速度向主题（topic）发布消息，节点2（node 2，命名listener）通过订阅者（subscriber）以10Hz速度订阅主题上的消息。

节点2共有三个订阅者，每个订阅者对应一个回调函数队列，其中订阅者1的回调函数队列通过ros::spinOnce()触发，订阅者2和订阅者3对应同样的callbackqueue(cq\_2)，并都通过ros::AsyncSpinner方式触发。

开始时，三个订阅者（可以理解为角色）同时运行，回调函数在接收到消息后正常触发，当计数（timer）到100时，暂停订阅者2和订阅者3，当计数到200时，重启订阅者2和订阅者3。

**测试运行**：catkin\_make --pkg callback\_msg\_clear\_tutorials

roscore & rosrun callback\_msg\_clear\_tutorials talker

rosrun callback\_msg\_clear\_tutorials listener

五、实验结果

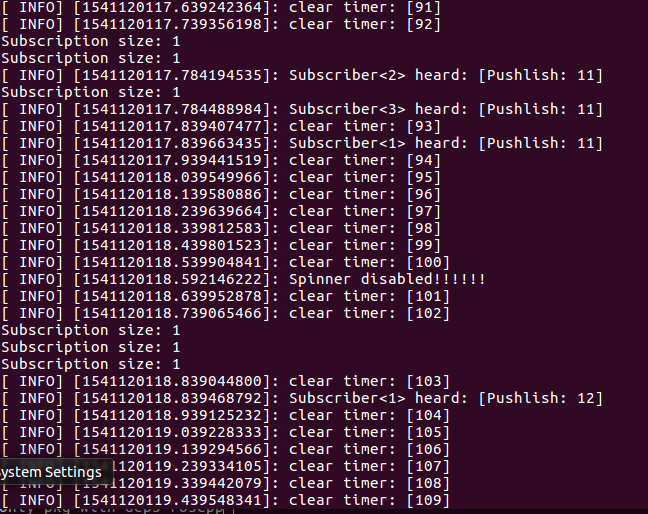
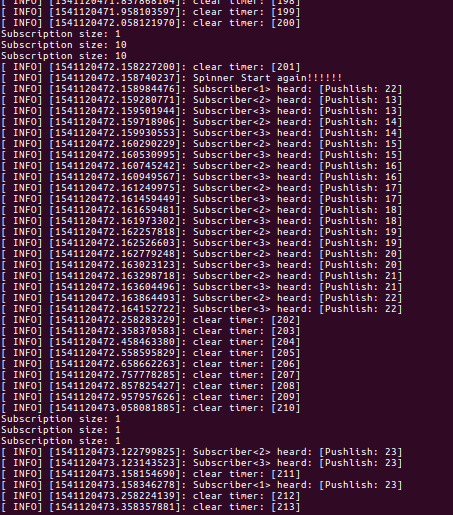
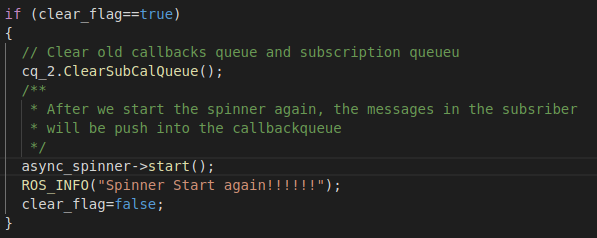
**1、源代码修改前**

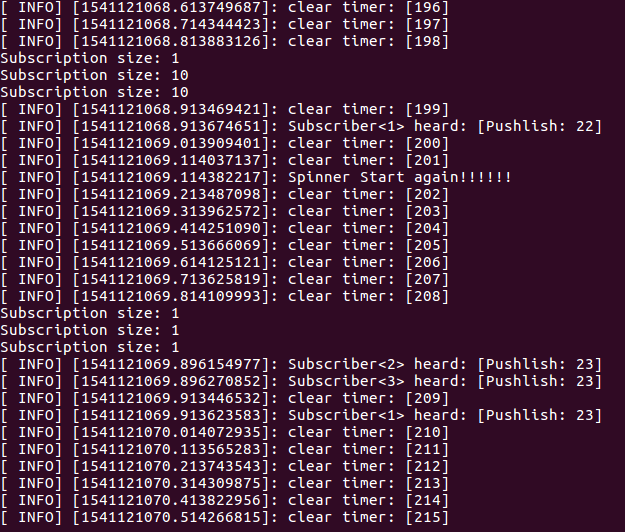
图1 修改源代码前，计数到100，暂停订阅者2和3的回调函数触发

如图1所示，当计数到100时，暂停订阅者2和订阅者3的回调函数触发（AsyncSpinner），订阅者2和订阅者3依然可以接收主题上的消息，消息队列长度增加，但回调函数不再触发调用。

图2 修改源代码前，计数到200，重启订阅者2和3的回调触发

如图2所示，当计数到200时，重启订阅者2和订阅者3的回调函数触发，回调函数首先调用处理消息队列中旧的消息。回调函数消息队列（cq\_2，长度为20）中依次存放对订阅者2消息队列（长度为10）和订阅者3消息队列（长度为10）的操作指针。

**2、源代码修改后**

图3 修改源代码后，计数到200，重启订阅者2的回调函数触发

如图3所示，当计数到200时，重启订阅者2的回调函数触发，重启前调用CallbackQueue类中的自定义函数ClearSubCalQueue()，对回调函数中的消息队列（size为20）及其对应的订阅者2（size为10）和订阅者3（size为10）的消息队列进行清空，重启后，直接调用回调函数处理当前消息。

附录1：函数继承示意图

**Callback\_queue\_interface.h**

**callback\_queue.h**

**subscription\_queue.h**

附录2：测试代码

