# ROS学习总结（4）- ROS node

## 节点创建

之前的文章属于铺垫部分，下面开始真正写ROS节点，文章参考ROS官网[1]和书[2]。之前我们提到节点Nodes通常有消息发布节点publish node和消息订阅节点subscribe node，发布节点会将消息发布到一个指定的主题topic上，订阅节点订阅该主题获取消息。同时发布与订阅的消息message类型需要事先定义。

首先我们进入到工作区里package的source文件夹中。

$ cd ~/learn\_ros\_ws/src/mastering\_ros\_demo\_pkg/src/

建立两个节点文件，**demo\_topic\_publisher.cpp** 和 **demo\_topic\_subscriber.cpp**。

发布节点文件定义如下：

#include "ros/ros.h"

#include "std\_msgs/Int32.h"

#include <iostream>

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

{

//初始化一个叫“demo\_topic\_publisher”的ROS节点

ros::init(argc, argv,"demo\_topic\_publisher");

//创建一个Nodehandle对象，用来与ROS进行通信

ros::NodeHandle node\_obj;

//创建一个topic发布节点，节点是number\_publisher，发布的topic的名称是"/numbers",发布的数据类型是Int32。第二项是buffer size参数。

ros::Publisher number\_publisher = node\_obj.advertise<std\_msgs::Int32>("/numbers",10);

//定义发送数据频率，如果频率高的话注意同时调高上一个buffer size

ros::Rate loop\_rate(10);

//初始化一个int变量

int number\_count = 0;

//当按Ctrl+C时，ros::ok()会返回0，退出该while循环，。

while (ros::ok())

{

//创建Int32类型的ROS message

std\_msgs::Int32 msg;

//将整数变量值赋给Message的data。data是msg对象的一个field。

msg.data = number\_count;

//Print出msg.data的值，同时也会存储在ROS的log系统里。

ROS\_INFO("%d",msg.data);

//节点发布message数据到topic，注意这里是msg。

number\_publisher.publish(msg);

//读取和更新ROS topics，如果没有spinonce()或spin()，节点不会发布消息。

ros::spinOnce();

//为了达到之前定义的发送频率，需要一个delay时间。

loop\_rate.sleep();

//整数变量自加1

++number\_count;

}

return 0;

}

订阅节点文件如下：

#include "ros/ros.h"

#include "std\_msgs/Int32.h"

#include <iostream>

//创建一个callback函数，当有新数据在topic更新时执行该函数，抽取新数据并print到console。

void number\_callback(const std\_msgs::Int32::ConstPtr& msg) {

ROS\_INFO("Received [%d]",msg->data);

}

int main(int argc, char \*\*argv) {

ros::init(argc, argv,"demo\_topic\_subscriber");

ros::NodeHandle node\_obj;

//创建一个ROS订阅节点number\_subscriber，订阅主题是"/numbers",第三项是callback函数。

ros::Subscriber number\_subscriber = node\_obj.subscribe("/numbers",10,number\_callback);

//写入spin()，相当于一个无限循环，订阅节点会等待topic更新数据，只有当按Ctrl+C才会退出。

ros::spin();

return 0;

}

## Build节点

我们需要编辑package文件夹里的CMakelist.txt文件才能编译和构建源代码。进入package文件夹找到CMakeLists.txt文件。加入如下信息：

include\_directories(

include

${catkin\_INCLUDE\_DIRS}

${Boost\_INCLUDE\_DIRS}

)

#This will create executables of the nodes

add\_executable(demo\_topic\_publisher src/demo\_topic\_publisher.cpp)

add\_executable(demo\_topic\_subscriber src/demo\_topic\_subscriber.cpp)

#This will generate message header file before building the target

add\_dependencies(demo\_topic\_publisher mastering\_ros\_demo\_pkg\_generate\_messages\_cpp)

add\_dependencies(demo\_topic\_subscriber mastering\_ros\_demo\_pkg\_generate\_messages\_cpp)

#This will link executables to the appropriate libraries

target\_link\_libraries(demo\_topic\_publisher ${catkin\_LIBRARIES})

target\_link\_libraries(demo\_topic\_subscriber ${catkin\_LIBRARIES})

我们注意到每个节点分别对应了executable，depedencies和target\_link，注意节点名字与文件，库的对应。

完成后，回到工作区路径并build示例package。

$ cd ~/learn\_ros\_ws/

$ catkin\_make

当构建结束后，启动ROS master，并在新的终端中允许两个节点程序。

$ roscore

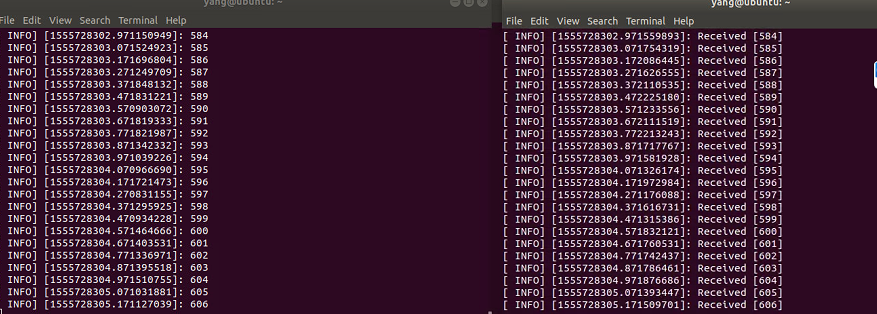
$ rosrun mastering\_ros\_demo\_pkg demo\_topic\_publisher

$ rosrun mastering\_ros\_demo\_pkg demo\_topic\_subscriber

注意如果在~/.bashrc中没有加入工作区的setup文件路径的话，需要在新开启的终端中再source一遍，否则会出现找不到package的情况。

$ source ~/learn\_ros\_ws/devel/setup.bash

之后运行发布节点和订阅节点的终端会显示如下

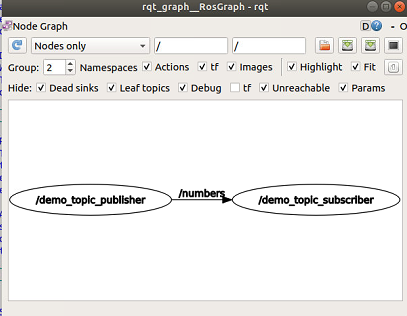


图表 1 - 终端输出

我们可以使用命令rqt\_graph来观察当前运行节点。

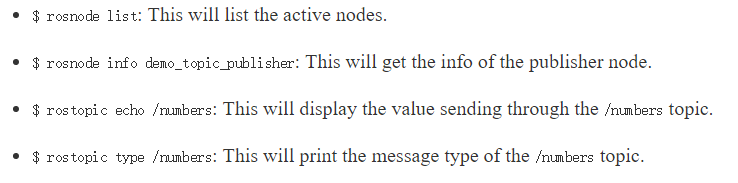
$ rqt\_graph

在新终端输入rqt\_graph后，得到结果如图2，



图表 2 - 运行节点图

其他指令如rosnode和rostopic可以用来debug和监视节点运行状态 [2]。



## 参考资料：

[1] *Documentation - ROS Wiki* (no date a). Available at: http://wiki.ros.org/ (Accessed: 12 March 2019).

[2] Joseph, L. and Cacace, J. (no date b) *Mastering ROS for robotics programming : design, build, and simulate complex robots using Robot Operating System*. Available at: https://learning.oreilly.com/library/view/mastering-ros-for/9781788478953/0efba8c8-ab2a-46f4-bc6c-672d2faf76fc.xhtml (Accessed: 17 April 2019).