**内部资料，注意保密**

# ROS机器人开发技术

# 中国大学MOOC

2018年7月V1.0



**中科重德智能有限公司**

目录

[ROS机器人开发技术 1](#_Toc1419087796)

[中国大学MOOC 1](#_Toc1777980039)

[前言 1](#_Toc1773736479)

[课程规划 1](#_Toc1393072164)

[子任务1——DEMO实例完整演示 2](#_Toc160999445)

[步骤 2](#_Toc111722151)

[子任务2——tf中常用的坐标转换 6](#_Toc724158384)

[背景知识 6](#_Toc1088122583)

[步骤 6](#_Toc703117860)

[子任务3——建立一个broadcaster 7](#_Toc85260988)

[背景介绍 8](#_Toc1053026638)

[回顾 8](#_Toc373626353)

[步骤 8](#_Toc386054275)

[结果检验 8](#_Toc724403305)

[子任务4——创建DEMO中的broadcaster 9](#_Toc1660201218)

[步骤 9](#_Toc1047136141)

[检验结果 11](#_Toc642152743)

# 前言

在学习完TF授课的相关内容后，我们已经对于TF有了一个直观地印象，它是用来进行坐标变换和维护坐标系关系的工具，而机器人的运动，可以看成是一个大的tf tree在实时的变化。

有了这种框架之后，我们开始对于TF涉及到的命令和使用场合进行更深一步的练习，以实现真正的可以使用TF工具。本着多练习，重复练习的态度，对于课时内容的设置，偏向于简单内容多练和基础知识上进行更深层次的练习使用。

# 课程规划

本次实训课分为三部分，第一部分是对于授课中的DEMO实例的再一次演示，同时对于授课中涉及的相关TF类库和TF工具都进行详细的解释和演示，在DEMO演示完成后，由同学们自己进行操作，对可能遇到的问题进行解答；第二部分是使用授课中提到的tf工具，具体为对于矩阵的函数，这一部分将是所有tf内容的基础；第三部分是使用broadcaster的类函数建立一个broadcaster，这是对于broadcaster的最简单的运用，之后在这个基础上，将复建DEMO实例中的broadcaster。

# 子任务1——DEMO实例完整演示

步骤：

NO.1 安装DEMO所需的环境，命令：

$ sudo apt-get install ros-kinetic-ros-tutorials ros-kinetic-geometry-tutorials ros-kinetic-rviz ros-kinetic-rosbash ros-kinetic-rqt-tf-tree

结果显示：

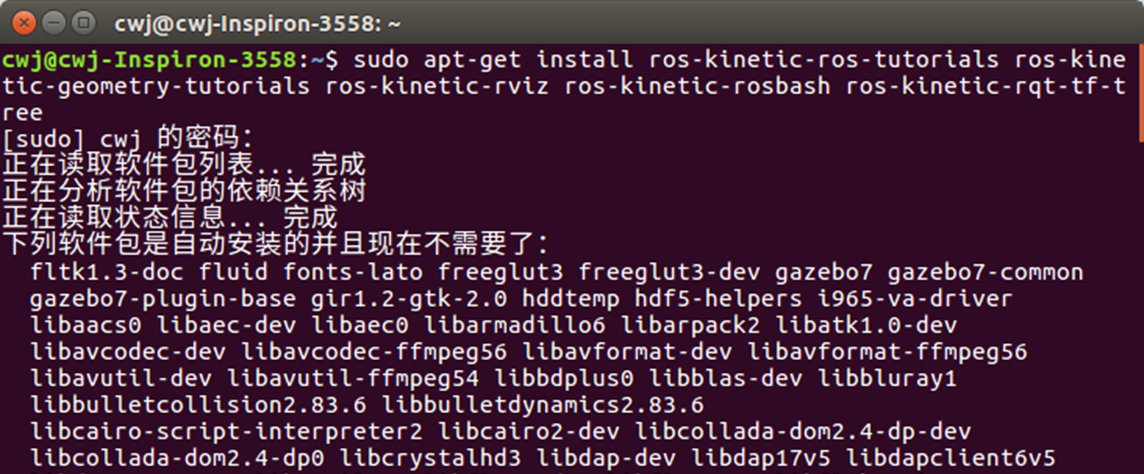


图1-1 安装DEMO显示图

NO.2 创建教程所需的pacakge，运行DEMO，命令：

$ roslaunch turtle\_tf turtle\_tf\_demo.launch

结果显示：

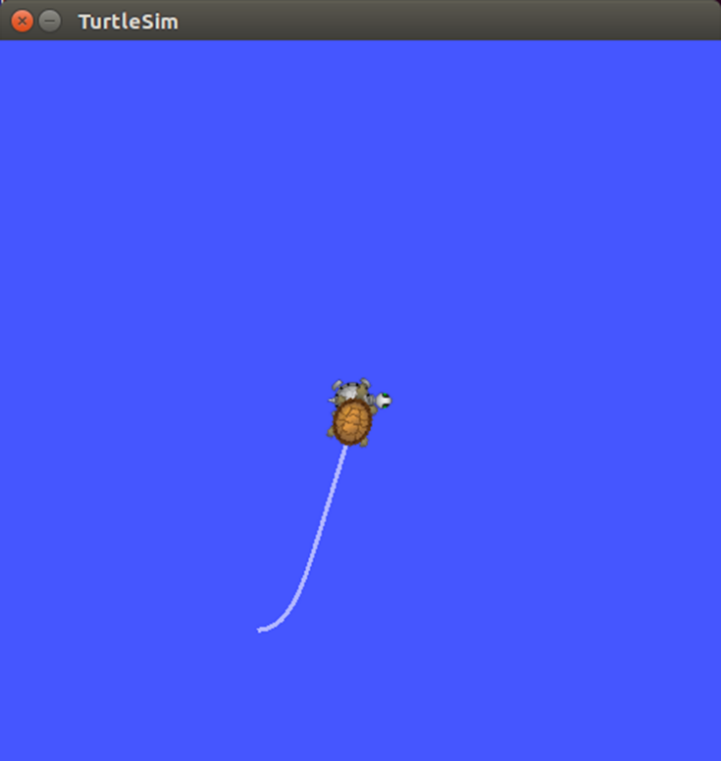


图1-2 DEMO运行初始图

注：驱动中心位置的小乌龟移动，需要选中“终端窗口（terminal）”才可以用方向键驱动小乌龟移动；这里注意留意授课中所提到的，是否在小乌龟背向或者侧向你的时候，控制它的方向偶尔会出错，如果你想像自己在坐着这只乌龟四处移动，那么情况会不会改变？驱动后的可能情况：

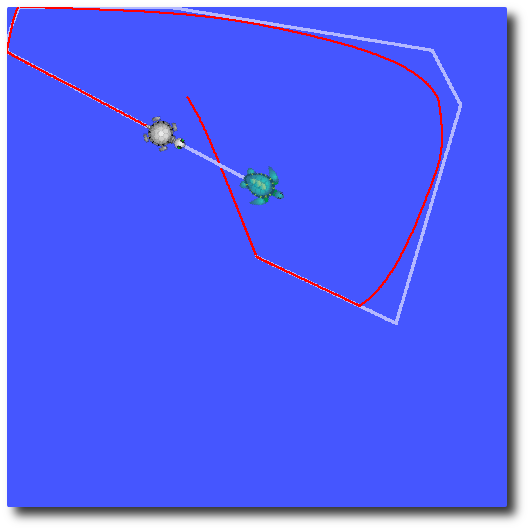


图1-3 DEMO运行结果图

正如第一节授课所提到的，这里tf库创建了三个坐标系：世界坐标系、turtle1坐标系和turtle2坐标系，并使用**tf broadcaster**来发布乌龟的坐标位置，用**tf listener**计算两个乌龟的坐标差，驱动一个乌龟“追赶”另一个只乌龟。

NO.3 使用tf tools查看详细的信息

View\_frames: 创建一个正在广播中的坐标系树图

命令：$ rosrun tf view\_frames

结果显示：

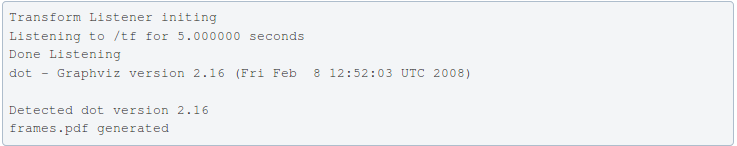


图1-4 view\_frame生成的信息图

可以看到有一个tf listener正在监听广播中的坐标系并将坐标系的关系用pdf文件生成了，查看这个树图可以到对应路径下，也可以通过命令$ evince frames.pdf直接查看，结果显示：见图

通过这张图，上面提到的三个坐标系就被展示出来，它们的连接关系也很清楚，world frame是turtle1 frame和turtle2 frame的父坐标系。这个命令对于调试也非常适用，可以显示出接收到的最近和最远的坐标系转换信息，和这些信息的广播速度。

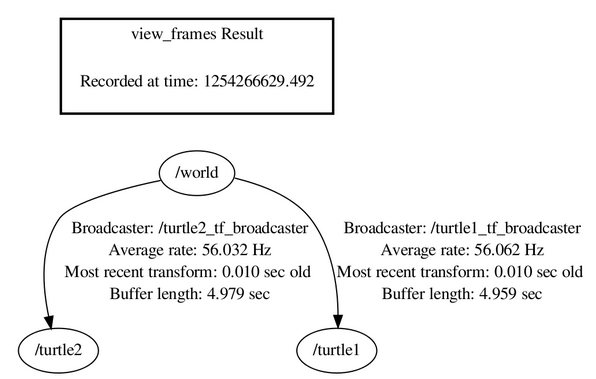
Rqt\_tf\_tree: 相似又不同于view\_frame，rqt\_tf\_tree命令是一个实时工具，可视化的将正

图1-5 view\_frame生成的tf tree图

在广播的坐标树信息显示出来，也可以通过左上角的刷新按钮，更新树图关系。

结果显示：

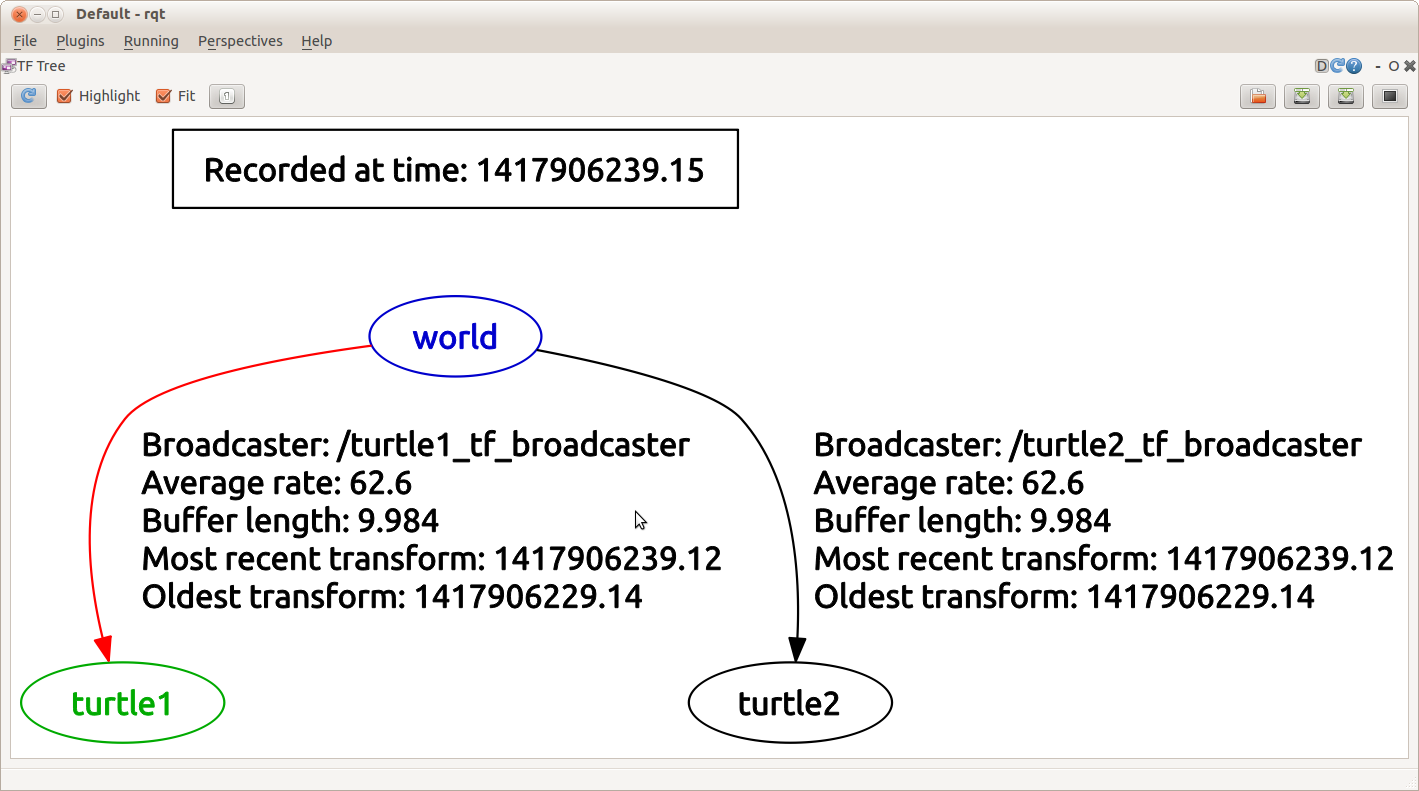


图1-6 rqt\_tf\_tree运行的结果示意图

Tf\_echo：返回（报告）正在ROS上广播的任意两个坐标系转换

格式：rosrun tf tf\_echo [reference\_frame] [target\_frame]

命令：$rosrun tf tf\_echo turtle1 turtle2

结果显示：正在ROS上广播的坐标系转换，驱动小乌龟移动，将会发现转换的对应变化。

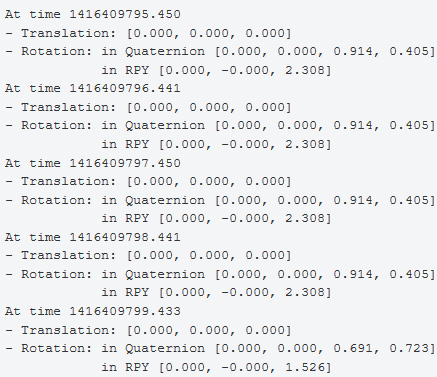


图1-7 tf\_echo结果图

Rviz & tf ：

命令：$ rosrun rviz rviz -d `rospack find turtle\_tf`/rviz/turtle\_rviz.rviz

注：rospack find turtle tf使用了英文的[ **’** ]符号括起来。

作用：使坐标系在三维图形中显示

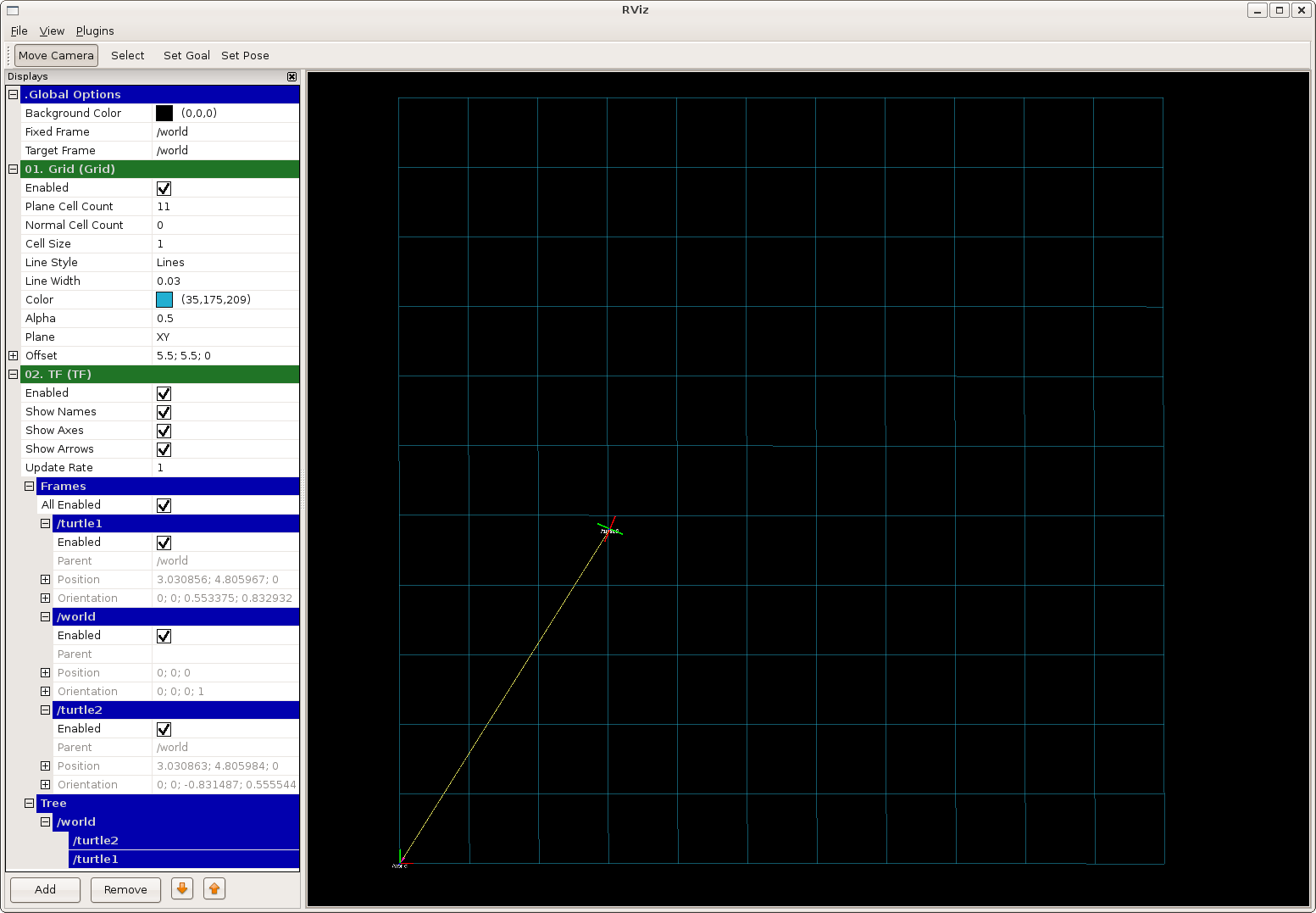


图1-8 rviz-d的DEMO结果图

驱动乌龟移动，可以看到对应的坐标系中的运动情况。

# 子任务2——tf中常用的坐标转换

背景知识

这里的坐标转换都是在授课都提到过的，这些坐标转换比较抽象，牵扯到的矩阵运算，欧拉角和四元数等数学知识，且不直观，但是都是机器人运动控制的基础，当学习完urdf之后，坐标系的概念会呈现出三维的立体图。这里只需要学会如何使用这些命令实现坐标矩阵的转换即可。

任务：尝试用rospy建立坐标转换的文件

步骤

NO.1 在你的package路径下新建一个名为script的文件夹，这个文件夹习惯上用来存放Python的源文件，这里在路径为/catkin\_ws/src/tf\_demo/script，在该路径下创建一个.py文件，名为py\_coordinate\_transformation.py，也可以使用其他名称，但推荐意义比较明确的名称，用gedit或roboware等任意一种编辑器打开创建的文件。

NO.2 首先是引入所需要的接口库，在文件开头输入

import rospy

import math

import tf

NO.3 初始化节点，向master注册名为py\_coordinate\_transformation。

NO.4 使用tf.transformations.random\_quaternion(rand=None)返回一个均匀随机四元数，方法是在文件中输入： q=tf.transformations.random\_quaternion(rand=None)

print '定义均匀随机四元数：'

print q

保存后在终端运行roscore，然后新开一个终端运行roscd %对应的package路径，这里是/catkin\_ws/src/tf\_demo，运行rosrun tf\_demo py\_coordinate\_transforma- tion .py，可以查看结果。

NO.5 按照相似的方法可依次输入以下的代码，参看对应的输出效果。

#1.2 返回均匀随机单位旋转矩阵

m=tf.transformations.random\_rotation\_matrix(rand=None)

print '定义均匀随机单位旋转矩阵：'

print m

#1.3 返回均匀随机单位向量

v=tf.transformations.random\_vector(3)

print '定义均匀随机单位向量：'

print v

#1.4 通过向量来求旋转矩阵

v\_m=tf.transformations.translation\_matrix(v)

print '通过向量来求旋转矩阵：'

print v\_m

#1.5 通过旋转矩阵来求向量

m\_v=tf.transformations.translation\_from\_matrix(m)

print '通过旋转矩阵来求向量：'

print m\_v

#第2部分，定义四元数

print '第2部分，定义四元数'

#2.1 通过旋转轴和旋转角返回四元数

axis\_q=tf.transformations.quaternion\_about\_axis(0.123, (1, 0, 0))

print '通过旋转轴和旋转角返回四元数：'

print axis\_q

#2.2 返回四元数的共轭

n\_q=tf.transformations.quaternion\_conjugate(q)

print '返回四元数q的共轭：'

print n\_q

#2.3 从欧拉角和旋转轴，求四元数

o\_q=tf.transformations.quaternion\_from\_euler(1, 2, 3, 'ryxz')

print '从欧拉角和旋转轴，求四元数：'

print o\_q

#2.4 从旋转矩阵中，返回四元数

m\_q=tf.transformations.quaternion\_from\_matrix(m)

print '从旋转矩阵中，返回四元数：'

print m\_q

#2.5 两个四元数相乘

qxq=tf.transformations.quaternion\_multiply(q,n\_q)

print '两个四元数相乘'

print qxq

#第3部分，定义欧拉角度

print '第3部分，定义欧拉角'

#3.1 从欧拉角和旋转轴返回旋转矩阵

rpy\_m=tf.transformations.euler\_matrix(1, 2, 3, 'syxz')

print '从欧拉角和旋转轴返回旋转矩阵'

print rpy\_m

#3.2 由旋转矩阵和特定的旋转轴返回欧拉角

m\_rpy=tf.transformations.euler\_from\_matrix(m)

print '由旋转矩阵和特定的旋转轴返回欧拉角'

print m\_rpy

#3.3 由四元数和特定的轴得到欧拉角

q\_rpy=tf.transformations.euler\_from\_quaternion(q)

print '由四元数和特定的轴得到欧拉角'

print q\_rpy

# 子任务3——建立一个broadcaster

## 背景介绍

Broadcaster类函数担任着向tf发布消息的角色，在授课中已经提及到它的两大类，即sendTransform()和sendTransformmessage()，这一任务中，将使用前者发布topic，对于后者，留到作业中练习使用sendTransformmessage()替换sendTransform()，这两个函数功能相似，格式不同而已。

## 回顾

sendTransform()函数的格式为sendTransform(translation,rotation,time,child,parent)，含义是把transform的平移(translation)和旋转(rotation)填好，打上时间戳(time)，然后表示出从父到子的frame流，发向/tf的topic。

## 步骤

NO.1 与上一任务相似部分不再重复，含义相同；

NO.2 输入br=tf.TransformBroadcaster() #br为broadcaster结构变量；

NO.3 输入相对原点的值和欧拉角，发布base\_link到link1的平移和旋转

x=1.0 #这一部分可以自己改

y=2.0

z=3.0

roll=0

pitch=0

yaw=1.57

rate = rospy.Rate(1)

while not rospy.is\_shutdown():

yaw=yaw+0.1

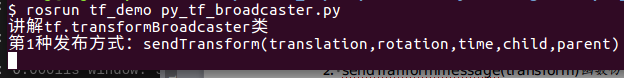
br.sendTransform((x,y,z), tf.transformations. quaternion\_from\_ euler(roll,pitch,yaw),rospy.Time.now(), "base\_link", "link1") #发布base\_link到link1的平移和翻转

rate.sleep()

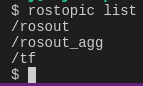
## 结果检验

这里我们发布了一个平移和翻转的信息，但是输出结果并没有直观的显示，所以我们想到了使用rostopic命令，查看是不是真的在向/tf发送topic

先看一下运行后的结果

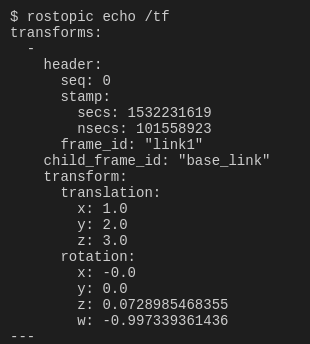


使用rostopic list命令列出正在发布的topic



查看/tf的信息，判断是否正在发布

使用rostopic echo /tf，可以看到结果



这里就可以确定sendTransform()确实发送了命令。

# 子任务4——创建DEMO中的broadcaster

步骤：

NO.1 创建一个新的的rospackage，在你想要的目录下创建一个名为learning\_tf的package,创建命令：$ cd %YOUR\_CATKIN\_WORKSPACE\_HOME%/src

$ catkin\_create\_pkg learning\_tf tf roscpp rospy turtlesimage，

注：其中的%YOUR\_CATKIN\_WORKSPACE\_HOME%是你选择的创建路径，可以用完整的路径名代替，如/home/Big\_apple/ROS/learning\_tf,也可以使用~/learning\_tf

NO.2 构建catkin

命令：$ cd %YOUR\_CATKIN\_WORKSPACE\_HOME%/

$ catkin\_make

$ source ./devel/setup.bash

以上两步为前期的准备工作，下面开始创建broadcaster

NO.3 创建源文件，进入创建的package路径

命令：$ roscd learning\_tf

在learning\_tf package中创建新的名为nodes的目录

命令：

$ mkdir nodes

NO.4在刚创建的nodes文件夹下创建一个名为turtle\_tf\_broadcaster.py的文件，这将创建broadcaster的文件，用编辑器打开这个文件，一般为gedit，均可，将下面的代码粘贴，保存。

Codes：

#!/usr/bin/env python

import roslib

roslib.load\_manifest('learning\_tf')

import rospy

import tf

import turtlesim.msg

def handle\_turtle\_pose(msg, turtlename):

br = tf.TransformBroadcaster()

br.sendTransform((msg.x, msg.y, 0),

tf.transformations.quaternion\_from\_euler(0, 0, msg.theta),

rospy.Time.now(),

turtlename,

"world")

if \_\_name\_\_ == '\_\_main\_\_':

rospy.init\_node('turtle\_tf\_broadcaster')

turtlename = rospy.get\_param('~turtle')

rospy.Subscriber('/%s/pose' % turtlename,

turtlesim.msg.Pose,

handle\_turtle\_pose,

turtlename)

rospy.spin()

注：在终端运行命令：$chmod +x nodes/turtle\_tf\_broadcaster.py使该文件可以运行。

NO.5 运行代码

在package路径下创建一个名为launch的文件夹，在该文件夹下创建名为start\_demo.launch的文件，并用编辑器打开，将下面代码粘贴，保存。

Codes：

<launch>

<!-- Turtlesim Node-->

<node pkg="turtlesim" type="turtlesim\_node" name="sim"/>

<node pkg="turtlesim" type="turtle\_teleop\_key" name="teleop" output="screen"/>

<node name="turtle1\_tf\_broadcaster" pkg="learning\_tf" type="turtle\_tf\_broadcaster.py" respawn="false" output="screen" >

<param name="turtle" type="string" value="turtle1" />

</node>

<node name="turtle2\_tf\_broadcaster" pkg="learning\_tf" type="turtle\_tf\_broadcaster.py" respawn="false" output="screen" >

<param name="turtle" type="string" value="turtle2" />

</node>

</launch>

NO.6运行你的broadcaster

命令：$ roslaunch learning\_tf start\_demo.launch

显示结果：

## 检验结果

使用tf\_echo命令检验turtle的位置是不是被广播到了tf

命令：$ rosrun tf tf\_echo /world /turtle1

结果显示

核心命令解释

turtlename = rospy.get\_param('~turtle')

解释：单参数节点，指定一个乌龟的名称，比如turtle1或turtle2

rospy.Subscriber('/%s/pose' % turtlename, turtlesim.msg.Pose, handle\_turtle\_pose, turtlename)

解释：节点订阅主题“turtleX / pose”并在每个传入消息上运行函数handle\_turtle\_pose

br = tf.TransformBroadcaster()

br.sendTransform((msg.x, msg.y, 0),

tf.transformations.quaternion\_from\_euler(0, 0, msg.theta),

rospy.Time.now(),

turtlename,

"world")

解释：发送乌龟的位置消息和从世界坐标系向turtleX转换的关系

作业：

1. 使用sendTranformmessage(transform)函数仿写任务三中的发送命令。

提示：需要使用到geometry\_msgs.msg模块

参考答案：见源码中的py\_tf\_broadcaster2.py

import rospy

import geometry\_msgs.msg

import tf2\_ros.transform\_broadcaster

import math

import tf

if \_\_name\_\_ == '\_\_main\_\_':

rospy.init\_node('py\_tf\_broadcaster')

print '讲解tf.transformBroadcaster类'

print '第2种发布方式：sendTransformMessage(transform)'

#第二部分，发布sendTransformMessage(transform)

m=tf.TransformBroadcaster()

t = geometry\_msgs.msg.TransformStamped()

t.header.frame\_id = 'base\_link'

t.header.stamp = rospy.Time(0)

t.child\_frame\_id = 'link1'

t.transform.translation.x = 1

t.transform.translation.y = 2

t.transform.translation.z = 3

t.transform.rotation.w=1

t.transform.rotation.x=0

t.transform.rotation.y=0

t.transform.rotation.z=0

#输入相对原点的值和欧拉角

rate = rospy.Rate(1)

while not rospy.is\_shutdown():

m.sendTransformMessage(t)

rate.sleep()