# TF坐标变换

在ROS中，每个子坐标系通过广播将子坐标系与父坐标系的关系，由tf树维护，通过查询tf树获取任意两个坐标系之间的变换关系，下面的例子说明了该用法。

## 创建功能包

使用catkin\_create\_pkg创建功能包：

|  |
| --- |
| catkin\_create\_pkg tf\_test roscpp rospy std\_msgs geometry\_msgs tf turtlesim |

## 创建TF广播器

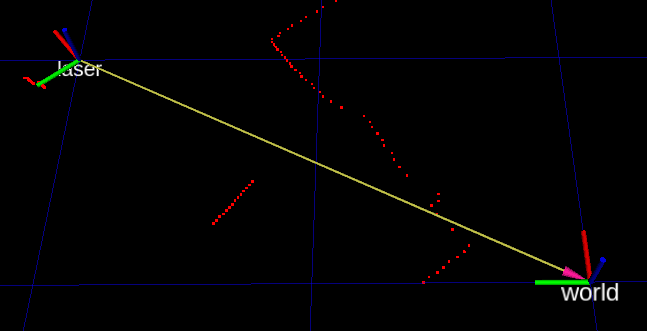
|  |
| --- |
| #include <ros/ros.h>  #include <tf/transform\_broadcaster.h>  #include <turtlesim/Pose.h>  std**::**string turtle\_name**;**  void poseCallback**(**const turtlesim**::**PoseConstPtr**&** msg**)**  **{**  static tf**::**TransformBroadcaster br**;**  tf**::**Transform transform**;**  transform**.**setOrigin**(**tf**::**Vector3**(**msg**->**x**,** msg**->**y**,** 0.0**));**  tf**::**Quaternion q**;**  q**.**setRPY**(**0**,** 0**,** msg**->**theta**);**  transform**.**setRotation**(**q**);**  br**.sendTransform(**tf**::**StampedTransform**(**transform**,** ros**::**Time**::**now**(),** "world"**,** turtle\_name**));**  **}**  int main**(**int argc**,** char**\*\*** argv**)**  **{**  ros**::**init**(**argc**,** argv**,** "tf\_broadcaster"**);**  **if(**argc **!=** 2**)**  **{**  ROS\_ERROR**(**"need turtle name as argument"**);**  **return** **-**1**;**  **}**  turtle\_name **=** argv**[**1**];**  ros**::**NodeHandle node**;**  ros**::**Subscriber sub **=** node**.**subscribe**(**turtle\_name **+** "/pose"**,** 10**,** **&**poseCallback**);**  ros**::**spin**();**  **return** 0**;**  **}** |

这里订阅了turtleX/pose消息，并设置poseCallback回调函数，在回调函数中根据乌龟的当前姿态设置其相对于世界坐标系的转换关系（turtleX的原点位置世界坐标系下(msg->x，msg->y，0)位置，与世界坐标系夹角(0, 0, msg->theta），并创建一个StampedTransform消息，广播出去，由tf树维护。

这里再详细说明一下br**.sendTransform(**tf**::**StampedTransform**(**transform**,** ros**::**Time**::**now**(),** "world"**,** turtle\_name**)**表示，turtle\_name这个坐标系由在world坐标系中的位置为tf**::**Vector3**(**msg**->**x**,** msg**->**y**,** 0.0**)**，角度为q**.**setRPY**(**0**,** 0**,** msg**->**theta**)**，用以下实验说明：

|  |
| --- |
| #include <ros/ros.h>  #include <tf/transform\_broadcaster.h>  #include <sensor\_msgs/LaserScan.h>  void laserCallback**(**sensor\_msgs**::**LaserScan msg**){**  static tf**::**TransformBroadcaster br**;**  tf**::**Transform transform**;**  transform**.**setOrigin**(** tf**::**Vector3**(**1.0**,** 2.0**,** 0.0**)** **);**  tf**::**Quaternion q**;**  q**.**setRPY**(**0**,** 0**,** 0.707**);**  transform**.**setRotation**(**q**);**  br**.**sendTransform**(**tf**::**StampedTransform**(**transform**,** ros**::**Time**::**now**(),**  "world"**,** "laser"**));**  **}**  int main**(**int argc**,** char**\*\*** argv**){**  ros**::**init**(**argc**,** argv**,** "add\_frame"**);**  ros**::**NodeHandle node**;**  ros**::**Subscriber sub **=** node**.**subscribe**(**"/scan"**,** 10**,** **&**laserCallback**);**  ros**::**spin**();**  **return** 0**;**  **};** |

上面的代码设置laser坐标系在world坐标中的位姿，即laser坐标系在world坐标系中(1.0, 2.0, 0.0)的位置，角度为逆时针旋转45度，我们可以从rviz中观察到该关系：



## 创建TF监听器

|  |
| --- |
| #include <ros/ros.h>  #include <tf/transform\_listener.h>  #include <geometry\_msgs/Twist.h>  #include <turtlesim/Spawn.h>  int main**(**int argc**,** char**\*\*** argv**)**  **{**  ros**::**init**(**argc**,** argv**,** "tf\_listener"**);**  ros**::**NodeHandle node**;**  ros**::**service**::**waitForService**(**"spawn"**);**  ros**::**ServiceClient add\_turtle **=** node**.**serviceClient**<**turtlesim**::**Spawn**>(**"spawn"**);**  turtlesim**::**Spawn srv**;**  add\_turtle**.**call**(**srv**);**  ros**::**Publisher turtle\_vel **=** node**.**advertise**<**geometry\_msgs**::**Twist**>(**"turtle2/cmd\_vel"**,** 10**);**  tf**::**TransformListener listener**;**  ros**::**Rate rate**(**10.0**);**  **while(**node**.**ok**())**  **{**  tf**::**StampedTransform transform**;**  **try**  **{**  listener**.waitForTransform(**"/turtle2"**,** "/turtle1"**,** ros**::**Time**(**0**),** ros**::**Duration**(**3.0**));**  listener**.lookupTransform(**"/turtle2"**,** "/turtle1"**,** ros**::**Time**(**0**),** transform**);**  **}**  **catch(**tf**::**TransformException**&** ex**)**  **{**  ROS\_ERROR**(**"%s"**,** ex**.**what**());**  ros**::**Duration**(**1.0**).**sleep**();**  **continue;**  **}**  geometry\_msgs**::**Twist vel\_msg**;**  vel\_msg**.**angular**.**z **=** 4.0 **\*** atan2**(**transform**.**getOrigin**().**y**(),** transform**.**getOrigin**().**x**());**  vel\_msg**.**linear**.**x **=** 0.5 **\*** sqrt**(**pow**(**transform**.**getOrigin**().**x**(),** 2**)** **+**  pow**(**transform**.**getOrigin**().**y**(),** 2**));**  turtle\_vel**.**publish**(**vel\_msg**);**  rate**.**sleep**();**  **}**  **return** 0**;**  **}** |

监听器通过waitForTransform和lookupTransform查询tf树上/turtle2和/turtle1的坐标关系transform，并根据该关系设置Twist消息发送到turtle2/cmd\_vel以驱动乌龟2执行跟随任务。

## launch文件

|  |
| --- |
| <launch>  <node pkg=**"turtlesim"** type=**"turtlesim\_node"** name=**"sim"**/>  <node pkg=**"turtlesim"** type=**"turtle\_teleop\_key"** name=**"teleop"** output=**"screen"**/>  <node pkg=**"tf\_test"** type=**"tf\_broadcaster"** args=**"/turtle1"** name=**"turtle1\_tf\_broadcaster"**/>  <node pkg=**"tf\_test"** type=**"tf\_broadcaster"** args=**"/turtle2"** name=**"turtle2\_tf\_broadcaster"**/>  <node pkg=**"tf\_test"** type=**"tf\_listener"** name=**"listener"**/>  </launch> |

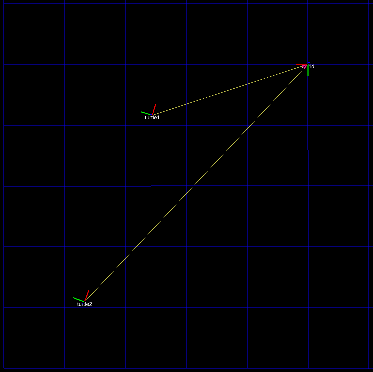
## CMakeLists.txt

在CMakeLists.txt中添加：

|  |
| --- |
| **add\_executable**(tf\_broadcaster src/tf\_broadcaster.cpp)  **target\_link\_libraries**(tf\_broadcaster ${catkin\_LIBRARIES})  **add\_executable**(tf\_listener src/tf\_listener.cpp)  **target\_link\_libraries**(tf\_listener ${catkin\_LIBRARIES}) |

## 运行

执行：roslaunch tf\_test tf.launch





运行rosrun tf view\_frames查看tf树：

