**中国大学慕课**

**实训文档**

**ROS机器人开发技术**

**实训5.1 topic in rospy**

|  |  |
| --- | --- |
| **姓名:** | **组号:05** |
| **实训负责人:** | **日期:2018-07-09** |

# 背景知识

ROS中的通信方式有四种,主题、服务、参数服务器、动作库。Topic是ROS里一种异步通信的模型，一般是节点间分工明确，有的只负责发送，有的只负责 接收处理。对于绝大多数的机器人应用场景，比如传感器数据收发，速度控制指令的收发， Topic模型是最适合的通信方式。话题需由用户开发，节点并不提供标准话题。包含消息源代码的文件存储在srv文件中。主题关联一个以包中.msg文件名称来命名的消息类型。

为了讲明白topic通信的编程思路，我们首先来看 topic\_demo 中的代码,这个程序是一个消息收 发的例子：自定义一个类型为gps的消息（包括位置x，y和工作状态state信息），一个node 以一定频率发布模拟的gps消息，另一个node接收并处理，算出到原点的距离。

主题发布和订阅编程流程

* 定义msg文件
* 创建发布者
* 创建订阅者
* 添加编译选项
* 运行可执行文件

# 任务1：创建ROS服务，通过Python编写一个简单的topic发布节点和订阅节点，并执行。

1. **创建一个topic\_rospy\_demo功能包，并创建一个服务。**

**1.1自定义消息的生成**

gps.msg 定义如下：

string state #工作状态

float32 x #x坐标

float32 y #y坐标

**1.2创建topic\_rospy\_demo功能包**

首先切换到catkin工作空间中的src目录下：

$ cd ~/catkin\_ws/src

现在使用catkin\_create\_pkg命令来创建一个名为'topic\_rospy\_demo'的功能包，这个程序包依赖于std\_msgs、roscpp和rospy：

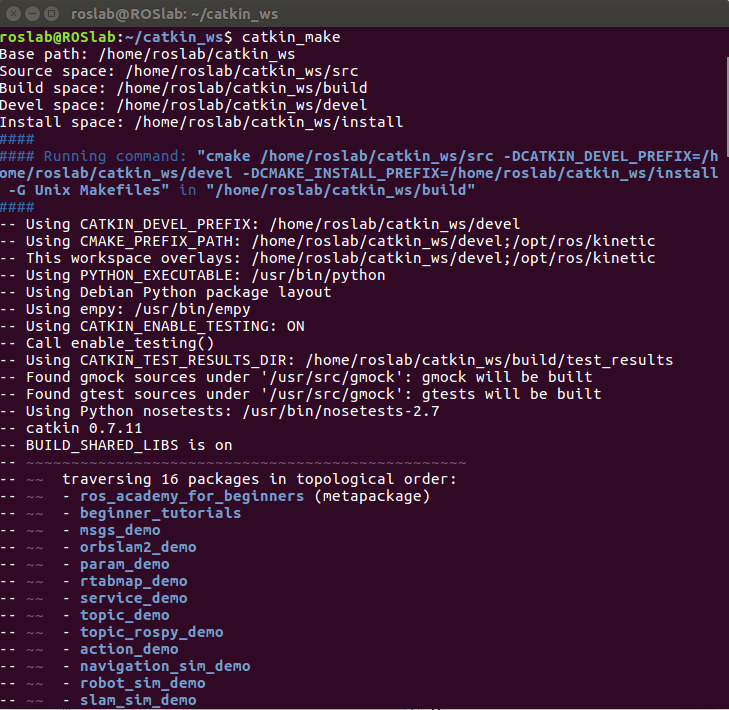
$catkin\_create\_pkg topic\_rospy\_demo std\_msgs rospy roscpp

这将会创建一个名为topic\_rospy\_demo的文件夹，这个文件夹里面包含一个[package.xml](http://wiki.ros.org/catkin/package.xml)文件和一个[CMakeLists.txt](http://wiki.ros.org/catkin/CMakeLists.txt)文件，这两个文件都已经自动包含了部分你在执行catkin\_create\_pkg命令时提供的信息。

回到catkin工作空间下，编译

$cd ..

$catkin\_make

得到如上图所示的结果则编译通过。

配置环境变量

$ source devel/setup.bash

**1.2创建一个话题topic**

下面，我们将在topic\_rospy\_demo 里定义新的服务。

首先用roscd命令进入包目录：

$ roscd topic\_rospy\_demo

创建一个srv目录

$ mkdir srv

此时我们将在catkin\_ws/src/topic\_rospy\_demo/文件夹下看到新建的msg空文件夹。

在srv文件夹里创建gps.msg文件，内容包括：

string state #工作状态

float32 x #x坐标

float32 y #y坐标

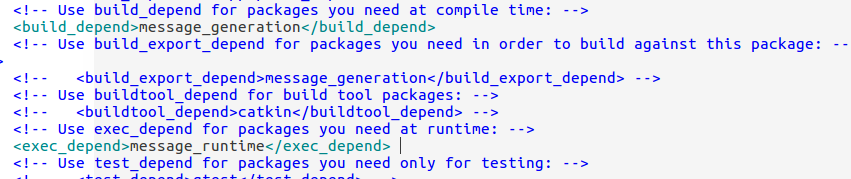
创建好之后，我们需要确保msg文件能够被转换为c++/python或者其他语言的代码。打开service\_rospy\_demo文件夹里的package.xml文件，

$ gedit package.xml

确保里面存在这两行且去掉它们的注释：

<!-- <build\_depend>message\_generation</build\_depend> -->

<!-- <exec\_depend>message\_runtime</exec\_depend> -->

在构建阶段我们需要 "message\_generation", 而在运行时我们需要 "message\_runtime"。

然后修改 CMakeLists.txt 文件。ROS的catkin编译系统会将你自定义的msg、srv(甚至还有action)文件自动编译构建,生成对应的C++、Python、LISP等语言下可用的库或模块。许多初学者错误地以为,只要建立了一个msg或srv文件,就可以直接在程序中使用,这是不对的,必须在 CMakeLists.txt 中添加关于消息创建、指定消息/服务文件那几个宏命令。

然后打开包目录下的CMakeLists.txt文件，在find\_package调用中添加message\_generation依赖，让你可以生成ROS信息。如下所示，括号里添加一项message\_generation即可：

find\_package(catkin REQUIRED COMPONENTS

roscpp

rospy

std\_msgs

message\_generation

)

你也许注意到有时你构建项目的时候，即使你没有在CMakeLists.txt中调用find\_package，仍然构建正常。这是因为catkin将你所有的项目集成到一起，只要你之前的项目调用了find\_package，现在的项目就会被配置为相同的调用。但是这是存在隐患的，当你独立构建一个项目的时候，缺少find\_package调用会导致构建失败。

然后同上面类似，确保你加入了message\_runtime 依赖，如图所示：

然后找到如下代码段：

去掉注释，改为：

add\_message\_files(

FILES

gps.msg

)

然后，确保generate\_messages()函数被调用：

去掉注释，修改为：

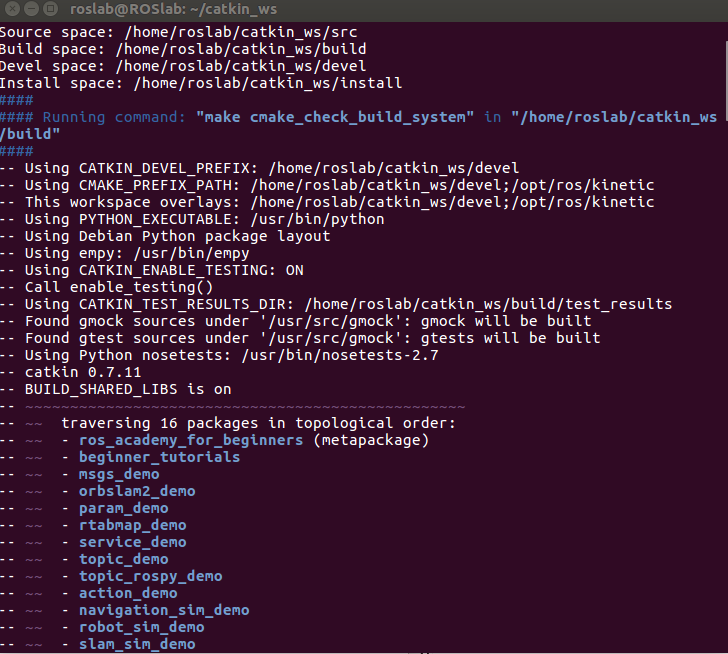
generate\_messages(

DEPENDENCIES

std\_msgs

)

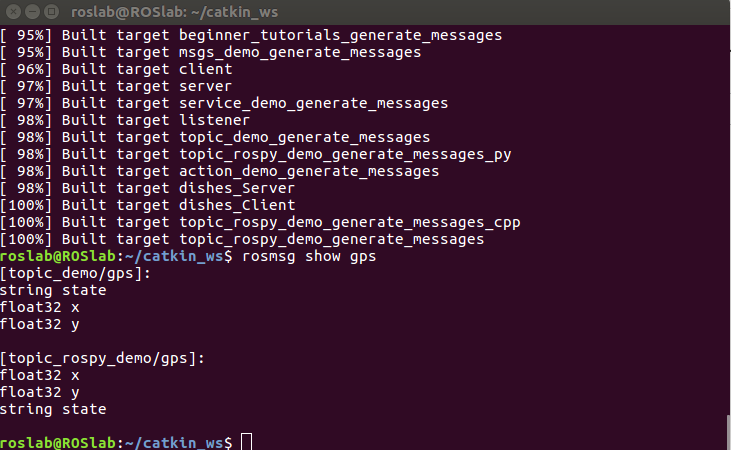
回到工作空间，catkin\_make一下，若结果如下图所示，则说明创建成功。



我们可以使用rossrv命令查看我们创建的服务的信息，确保它被ROS正确识别：

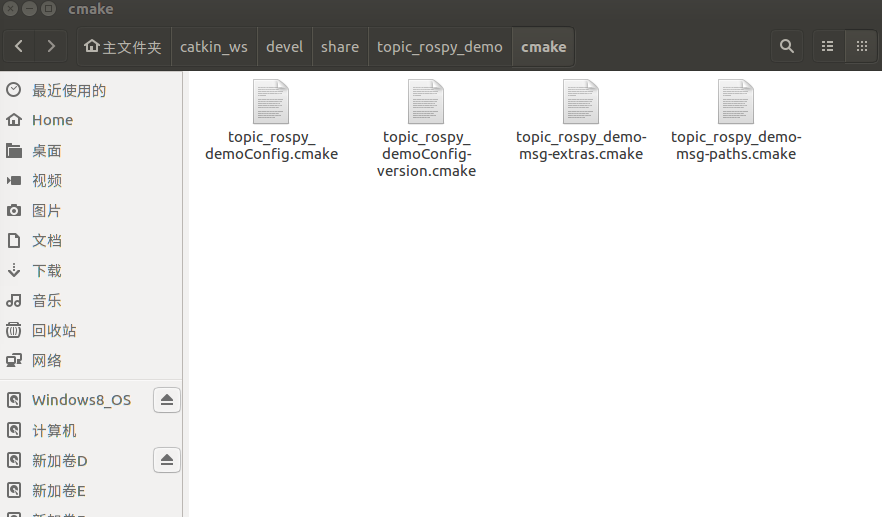
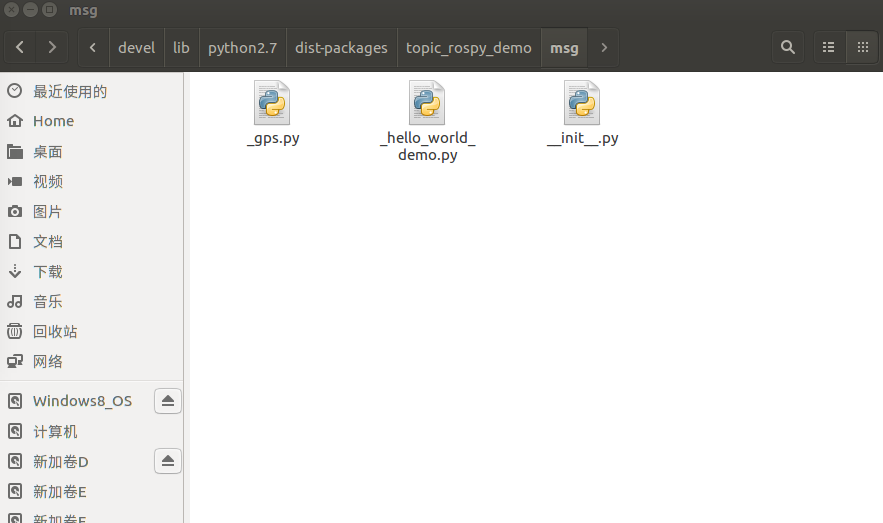
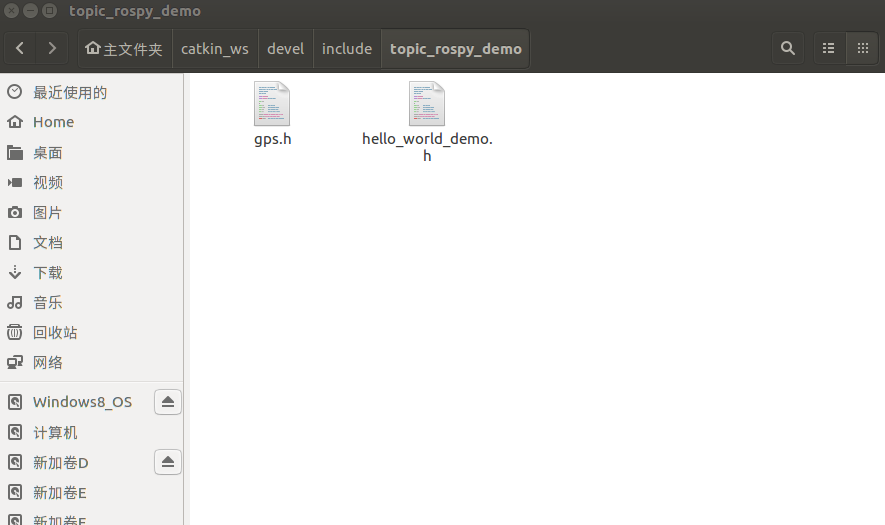
$ rosmsg show topic\_rospy\_demo/ gps.msg

得到



当然类似rosmsg，也可以省略包名 $ rosmsg show gps.msg

现在，srv目录下的所有.msg文件，都会生成ROS支持的语言的源代码，C++的头文件在~/catkin\_ws/devel/include/topic\_rospy\_demo/里，Python基本在 ~/catkin\_ws/devel/lib/python2.7/dist-packages/topic\_rospy\_demo/srv/里，lisp文件则在 ~/catkin\_ws/devel/share/common-lisp/ros/topic\_rospy\_demo/srv/ 里。



**2.创建话题发布节点（publisher）**

在topic\_rospy\_demo/src/文件下新建scripts文件夹，用于存放编写的Python脚本。在文件夹下创建pytalker.py文件。内容代码如下：

#!/usr/bin/env python

# coding:utf-8

# 上面指定编码utf-8，使python能够识别中文

import rospy

#导入自定义的数据类型

from topic\_demo.msg import gps

def talker():

#Publisher 函数第一个参数是话题名称，第二个参数 数据类型，现在就是我们定义的msg 最后一个是缓冲 区的大小

#queue\_size: None（不建议） #这将设置为阻塞式同步收发模式！

#queue\_size: 0（不建议）#这将设置为无限缓冲区模式，很危险！

#queue\_size: 10 or more #一般情况下，设为10 。queue\_size太大了会导致数据延迟不同步。 pub = rospy.Publisher('gps\_info', gps , queue\_size=10)

rospy.init\_node('pytalker', anonymous=True)

#更新频率是1hz

rate = rospy.Rate(1)

x=1.0

y=2.0

state='working'

while not rospy.is\_shutdown():

#计算距离

rospy.loginfo('Talker: GPS: x=%f ,y= %f',x,y)

pub.publish(gps(state,x,y))

x=1.03\*x

y=1.01\*y rate.sleep()

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

talker()

**3创建话题订阅节点（subscriber)**

在scripts文件夹下创建文件pylistener.py 。内容代码如下

#!/usr/bin/env python

# coding:utf-8

# 上面的第二句指定编码类型为utf-8，是为了使python能够识别中文

# 加载所需模块

import rospy

import math

#导入mgs

from topic\_demo.msg import gps

#回调函数输入的应该是msg

def callback(gps):

distance = math.sqrt(math.pow(gps.x, 2)+math.pow(gps.y, 2))

rospy.loginfo('Listener: GPS: distance=%f, state=%s', distance, gps.state)

def listener():

rospy.init\_node('pylistener', anonymous=True)

#Subscriber函数第一个参数是topic的名称，第二个参数是接受的数据类型 第三个参数是回调函数的名称

rospy.Subscriber('gps\_info', gps, callback)

rospy.spin()

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

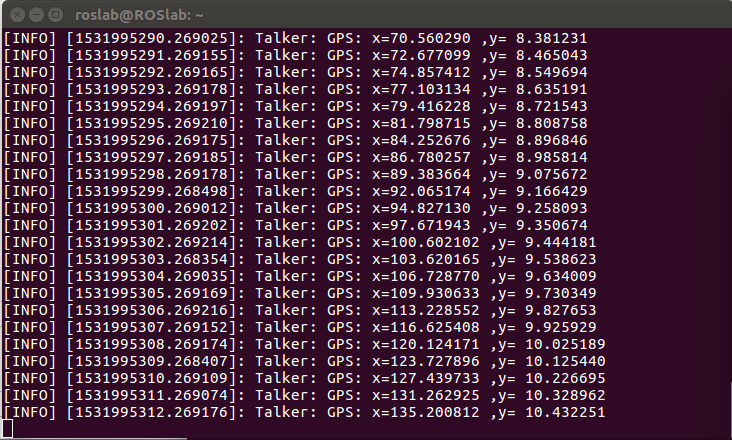
listener()

**4.测试**

打开终端 运行roscore

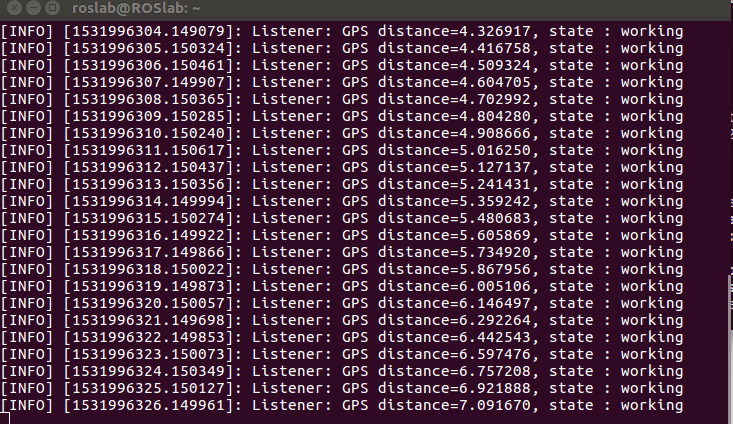
打开一个新的终端，运行发布节点

$ rosrun topic\_rospy\_demo pytalker.py



再打开一个新的终端，启动订阅端节点

$ rosrun topic\_rospy\_demo pylistener.py



成功。