在ROS中使用Python发布和订阅消息

[1 用Python实现ROS节点 1](#_Toc12547)

[2 注意事项 3](#_Toc10862)

[3 ROS中python和cpp混合编程 4](#_Toc26840)

[4 ROS中创建cpp工作空间 4](#_Toc4553)

[(1) 基本工作空间创立 4](#_Toc32219)

[(2) ROS\_work\_space\_package\_node\_topic的关系 5](#_Toc2724)

[(3) 编写cpp发布器和订阅器代码 5](#_Toc8251)

[(4) CMakeLists\_txt文件 6](#_Toc24270)

[确认原始CMakeLists.txt文件 6](#_Toc19330)

[在 CMakeLists.txt 文件末尾加入几条语句: 7](#_Toc22739)

[说明 7](#_Toc4230)

[(5) 编译及运行流程 7](#_Toc31913)

[① 进入catkin\_test项目文件夹的根目录； 7](#_Toc10238)

[② source ./devel/setup.bash 7](#_Toc6774)

[③ 输入catkin\_make命令，编译源码； 7](#_Toc23188)

[④ 设置环境变量： source ./devel/setup.bash 7](#_Toc13527)

[⑤ 启动ROS Master：roscore 7](#_Toc29354)

[5 ROS学习笔记rospy介绍 7](#_Toc15888)

[6 参考文献 12](#_Toc19758)

# 用Python实现ROS节点

ROS主要的编程语言不光是C++，也有Python。这里讲解如何利用Python实现ROS节点的编写。教程参考官网rospy\_tutorials。

1. 编写一个简单的发布者和订阅者

1创建工作空间

建立文件夹hello\_rospy,再在该目录下建立子目录src，cd到该src目录，运行如下命令创建工作包。

catkin\_create\_pkg beginner\_tutorials std\_msgs rospy roscpp

cd到src上层目录，编译并且source

cd ..

catkin\_make

source ./devel/setup.bash

2编写发布者程序和订阅者程序

roscd beginner\_tutorials/

mkdir scripts

cd scripts

在scripts目录下新建talker.py文件，填写如下内容：

#!/usr/bin/env python

# license removed for brevity

import rospy

from std\_msgs.msg import String

def talker():

pub = rospy.Publisher('chatter', String, queue\_size=10)

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

rate = rospy.Rate(10) # 10hz

while not rospy.is\_shutdown():

hello\_str = "hello world %s" % rospy.get\_time()

rospy.loginfo(hello\_str)

pub.publish(hello\_str)

rate.sleep()

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

try:

talker()

except rospy.ROSInterruptException:

pass在scripts目录下新建listener.py文件，填写如下内容：

#!/usr/bin/env python

import rospy

from std\_msgs.msg import String

def callback(data):

rospy.loginfo(rospy.get\_caller\_id() + "I heard %s", data.data)

def listener():

# In ROS, nodes are uniquely named. If two nodes with the same

# node are launched, the previous one is kicked off. The

# anonymous=True flag means that rospy will choose a unique

# name for our 'listener' node so that multiple listeners can

# run simultaneously.

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

rospy.Subscriber("chatter", String, callback)

# spin() simply keeps python from exiting until this node is stopped

rospy.spin()

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

listener()

记得更改权限为可执行文件

3. 编译

修改Cmakelist.txt为如下：

cmake\_minimum\_required(VERSION 2.8.3)

project(beginner\_tutorials)

## Find catkin macros and libraries

## if COMPONENTS list like find\_package(catkin REQUIRED COMPONENTS xyz)

## is used, also find other catkin packages

find\_package(catkin REQUIRED COMPONENTS

roscpp

rospy

std\_msgs

)

catkin\_package()

直接在工作空间运行catkin\_make即可。

4. 运行

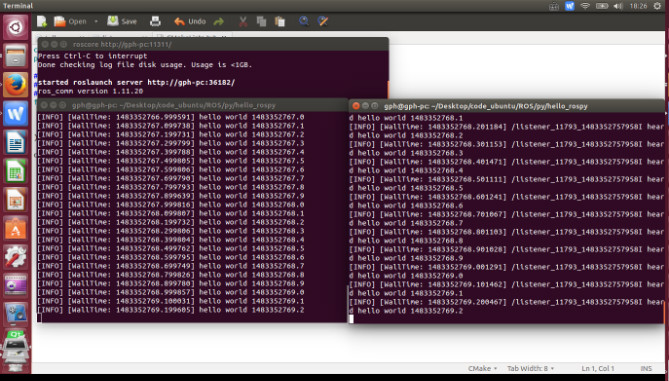
在3个终端中分别执行如下命令

roscore

rosrun beginner\_tutorials talker.py

rosrun beginner\_tutorials listener.py

结果如下：



# 注意事项

注意：py文件第一行必须加上 #!/usr/bin/env python3 否则无法运行代码。相当于解释器。

为test\_pub.py添加执行权限：

chmod +x test\_pub.py

然后修改cmakelists.txt进行编译。其实python是不需要修改cmakelists.txt进行编译的，因为python本身为脚本文件，无需编译，只需添加权限就能运行。下面在cmakelists.txt中加入这条是为了用launch启动这个文件时能够找到它，如果是用rosrun运行，不加也是可以的。

cmakelist.txt中加入：

catkin\_install\_python(PROGRAMS

scripts/test\_pub.py

DESTINATION ${CATKIN\_PACKAGE\_BIN\_DESTINATION}

)

**编译时注意：如果代码中需要用python3的一些库，如numpy1.19.5，pytorch1.7d等。就需要进行激活虚拟环境进行编译，否则会提示找不到包或者模块。**

**注意：在ros中编译python文件不会像cpp文件一样生成可执行文件。运行时还是：rosun 功能包名字 test\_pub.py。**

# ROS中python和cpp混合编程

在ROS中，可以同时运行python代码和cpp代码开发的ROS工作空间中的项目节点。但是这两个工作空间的项目需要分别独立开发。开发好之后运行时可以共用一个roscore命令来启动ROS中的主节点。由python代码开发的发布或订阅消息的节点文件，可以订阅cpp代码开发的节点所发布的消息。也可以由cpp代码开发的节点订阅python开发的节点所发布的topic。实现了两种语言开发节点的相互之间的主题订阅和发布。

# ROS中创建cpp工作空间

## 基本工作空间创立

参考<<[胡春旭. 2021-ROS机器人开发实践](https://www.baidu.com/link?url=RqEFkD376cM-ikOjeYZFgrLQSeJcmrX6t3RkwfAL04-k1uLqY67rXqX6tSqqbUCimjhT7e-K0suFbessM-QIoHRYaZGV9vZhAlkAC6haBOe&wd=&eqid=aac40035001a24af0000000462afd9ef" \t "https://www.baidu.com/_blank)>>section3.2部分。

原文链接：https://blog.csdn.net/qq\_24624539/article/details/102755619

安装完成ROS以后会有一个catkin\_ws文件夹，即为ROS工作空间。

[1]创建并初始化一个新的工作空间catkin\_ws

mkdir -p ~/catkin\_ws/src

cd ~/catkin\_ws/src

catkin\_init\_workspace

catkin\_init\_workspace命令把当前目录初始化为一个ROS工作空间。

[2]编译该工作空间。

cd ~/catkin\_ws

catkin\_make

[3]定义catkin\_ws空间所需要的环境变量。执行此命令后，ros的相关命令（如roscd等）可以找到此工作空间中的package。

source ~/catkin\_ws/devel/setup.bash

[4]验证ROS工作空间的环境变量加载成功：

echo $ROS\_PACKAGE\_PATH

[5] create function package

enter the src directory of function package:

catkin\_create\_pkg beginner\_tutorials std\_msgs rospy roscpp

enter the root directory:

catkin\_make

source ./devel/setup.bash

然后运行rosrun启动节点命令，或roslaunch批量化启动节点命令。

[5]打开环境变量文件sudo gedit .bashrc 加入下面一行：

source ~/catkin\_ws/devel/setup.bash

source ~/.bashrc

原文链接：<https://blog.csdn.net/weixin_43297891/article/details/115426161>

## ROS\_work\_space\_package\_node\_topic的关系

ROS中的工作空间（work space）是ROS中非常重要的一个概念，可以把工作空间理解为一个大的工厂，里面的分成几个大的生产车间（package），每一个生产车间中会有若干个具有不同技能的工人（node）。当工厂运转时，每个车间中的工人node同时工作，他们通过话题（topic）进行信息沟通。各个大的车间之间也存在这互相依赖的关系，共同组成一个有机的整体。

Node是指ROS中运行的最小处理单元。在ROS中，建议为一个目的创建一个节点，设计时注重可重用性。节点运行的同时，向主节点注册节点名称，并且还注册发布者(publisher)/订阅者(subscriber)，服务服务器(service server)/服务客户端(service client), 且注册消息形式，URI地址和端口。每个节点可以使用话题(topic)和服务(service)与其他节点交换消息。发布者(publisher)/订阅者(subscriber)之间传递topic，服务服务器(service server)/服务客户端(service client)之间传递service。

ROS node 之间的通讯形式主要包括两种：topic 和 service。

通过 topic 通讯时，不同的 node 可以向同一个 topic 上发送、接收数据，发送数据的 node 不知道数据是从哪个 node 发送过来的，同样地，发送数据的 node 也不知道是哪个 node 接收了数据。因此，每个 node 都是相对独立的，只需要负责自己的功能实现以及外部接口，不需要关心其他 node 的行为。这是一种开放式的收、发数据的方式，也是 node 之间通讯的主要形式，有利于构造分布式大系统。

service 则是一种请求+反馈的通信机制。消息的传输只涉及两个 node：发送请求的一方称为 client，提供服务的一方叫做 server。在通过 service 形式进行通讯时，client 首先向 server 请求服务, 收到消息之后 server 运行事先设置好的服务功能，并返回消息给 client。service 通讯一般用在事件触发情景中，例如满足某个条件就令 node 开启某项功能，并希望确认功能确实顺利开启。

通常情况下，一个cpp文件中定义一个节点(node)；但是在一个cpp文件中，可以在所定义的节点中，发布或订阅多条主题(topic)或一条主题(topic)。

https://blog.csdn.net/puqian13/article/details/91360102

<https://blog.csdn.net/qq_41972382/article/details/95209741>

## 编写cpp发布器和订阅器代码

初始化 ROS 系统

在 ROS 网络内广播我们将要在 chatter 话题上发布 [std\_msgs/String](https://link.zhihu.com/?target=http://docs.ros.org/api/std_msgs/html/msg/String.html" \t "https://zhuanlan.zhihu.com/p/_blank) 类型的消息

以每秒 10 次的频率在 chatter 上发布消息

在 beginner\_tutorials package 里创建 src/talker.cpp 文件：

#include <sstream>

#include "ros/ros.h"

#include "std\_msgs/String.h"

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

ros::init(argc, argv, "talker");

ros::NodeHandle n;

ros::Publisher chatter\_pub = n.advertise<std\_msgs::String>("chatter", 1000);

ros::Rate loop\_rate(10);

/\* a unique string for each message. \*/

int count = 0;

while (ros::ok()) //ros::ok()

{

std\_msgs::String msg;

std::stringstream ss;

ss << "hello world, I am a genius! " << count;

msg.data = ss.str();

ROS\_INFO("%s", msg.data.c\_str());

chatter\_pub.publish(msg);

ros::spinOnce();

loop\_rate.sleep();

++count;

}

return 0;}

在 beginner\_tutorials package 目录下创建 src/listener.cpp 文件：

#include "ros/ros.h"

#include "std\_msgs/String.h"

void chatterCallback(const std\_msgs::String::ConstPtr& msg)

//是一个回调函数，当接收到 chatter 话题的时候就会被调用。

{ ROS\_INFO("I heard: [%s]", msg->data.c\_str());

}

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

{ ros::init(argc, argv, "listener");

ros::NodeHandle n;

ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);

ros::spin();

return 0;

}

## [CMakeLists\_txt](https://link.zhihu.com/?target=http://wiki.ros.org/catkin/CMakeLists.txt" \t "https://zhuanlan.zhihu.com/p/_blank)文件

## 确认原始[CMakeLists.txt](https://link.zhihu.com/?target=http://wiki.ros.org/catkin/CMakeLists.txt" \t "https://zhuanlan.zhihu.com/p/_blank)文件

原始文件中保留的有用代码如下：

cmake\_minimum\_required(VERSION 2.8.3)

project(beginner\_tutorials)

## Find catkin and any catkin packages

find\_package(catkin REQUIRED COMPONENTS

roscpp

rospy

std\_msgs

genmsg)

catkin\_package()

## 在 [CMakeLists.txt](https://link.zhihu.com/?target=http://wiki.ros.org/catkin/CMakeLists.txt" \t "https://zhuanlan.zhihu.com/p/_blank) 文件末尾加入几条语句:

include\_directories(include ${catkin\_INCLUDE\_DIRS})

add\_executable(talker src/talker.cpp)

target\_link\_libraries(talker ${catkin\_LIBRARIES})

add\_dependencies(talker ${PROJECT\_NAME}\_generate\_messages\_cpp)

add\_executable(listener src/listener.cpp)

target\_link\_libraries(listener ${catkin\_LIBRARIES})

add\_dependencies(listener ${PROJECT\_NAME}\_generate\_messages\_cpp)

## 说明

1-2会生成两个可执行文件, talker 和 listener, 默认存储到 [devel space](https://link.zhihu.com/?target=http://wiki.ros.org/catkin/workspaces%23Development_.28Devel.29_Space" \t "https://zhuanlan.zhihu.com/p/_blank) 目录下,具体在~/catkin\_ws/devel/lib/<package name> 中.

3.如果在 \*Groovy\* 版本下，你可以使用下边的这个变量来添加对所有必须的文件依赖:

add\_dependencies(talker ${catkin\_EXPORTED\_TARGETS})

## 编译及运行流程

## 进入catkin\_test项目文件夹的根目录；

## source ./devel/setup.bash

## 输入catkin\_make命令，编译源码；

## 设置环境变量： source ./devel/setup.bash

## 启动ROS Master：roscore

# ROS学习笔记rospy介绍

rospy介绍（一）

rospy是Python版本的ROS客户端库，提供了Python程序需要的接口（rospy就是一个Python模块），位于/opt/ros/kineetic/lib/python2.7/dist-packages/rospy。

rospy和roscpp

rospy功能与roscpp相似，都有关于node、topic、service、param、time相关操作，也有一些区别：

rospy没有NodeHandle。

创建publisher、subscriber等操作被直接封装成了rospy中的函数或类，调用简单。

rospy部分接口命名和roscpp不同。

相较于C++开发，Python写ROS程序的开发效率较高，不用需要再注意显示、类型转换等细节，但执行效率低，同样一个功能用Python运行的耗时高于C++。所以开发SLAM、路径规划、机器视觉等方面算法时，往往优先选择C++。

ROS中绝大多数基本指令（rostopic、roslaunch等）都是用Python开发的，简单轻巧。

ROS中Python代码的组织方式

Python代码有两种组织方式：单独的Python脚本（适用于简单的程序），Python模块（适用于提体量较大的程序）。

单独的Python脚本：

小体量的ROS程序，一般就是一个Python文件，放在script/路径下

test(包名)/scripts/file\_name.py

1

Python模块

当程序的功能比较复杂时，无法放到一个脚本里，就要把一些功能放到PythonModule中，使用其他的脚本来调用。

ROS建立Python模块的规范：

test/src/test(与package同名)/init.py 可参考http://www.cnblogs.com/Lands-ljk/p/5880483.html

test/src/test(与package同名)/modulefiles.py

test/scripts/your\_script.py

test/scripts/setup.py

常用的ROS命令大多数都是Python模块，源码存放在ros\_comm仓库的tools路径下：https://github.com/ros/ros\_comm/tree/lunar-devel/tools

每一个命令行工具（如rosbag、rosmsg）都是用模块的形式组成核心代码，然后在script/下建立一个脚本来调用。

常用rospy的API

具体查看http://docs.ros.org/api/rospy/html/rospy-module.html

Node相关

返回值 方法 作用

rospy.init\_node(name,argv=None,anonymous=False) 注册和初始化node

MasterProxy rospy.get\_master() 获取master的句柄

bool rospy.is\_shoudow() 节点是否关闭

rospy.on\_shutdown() 在节点关闭时调用fn函数

str get\_node\_uri 返回节点的URI

str get\_name() 返回本节点的全名

str get\_namespace() 返回本节点的名字空间

… … …

Topic相关

函数：

返回值 方法 作用

[[str,str]] get\_published\_topics() 返回正在被发布的所有topic名称和类型

Message wait\_for\_message() 等待某个topic的message

spin() 触发topic或service的回调/处理函数，会阻塞直到关闭节点

… … …

Publisher类：

返回值 方法 作用

init(self,name,data\_class,queue\_size=None) 构造函数

publish(self,msg) 发布消息

str unregister(self) 停止发布

… … …

Subscriber类：

返回值 方法 作用

init\_(self,name,data\_class,call\_back=None,queue\_size=None) 构造函数

unregister(self,msg) 停止订阅

… … …

Service相关

函数：

返回值 方法 作用

wait\_for\_service(service,timeout=None) 阻塞直到服务可用

… … …

Service类(server)：

返回值 方法 作用

init(self,name,service\_class,handler) 构造函数，handler为处理函数，service\_class为srv类型

shutdown(self) 关闭服务的server

… … …

ServiceProxy类(client)：

返回值 方法 作用

init(self,name,service\_class) 构造函数，创建client

call(self,ars,\*kwds) 发起请求

call(self,args,\*kwds) 同上

close(self) 关闭服务的client

… … …

Param相关

函数：

返回值 方法 作用

XmlRpcLegalValue get\_param(param\_name,default=\_unspecified) 获取参数的值

[str] get\_param\_names() 获取参数的名称

set\_param(param\_name,param\_value) 设置参数的值

delete\_param(param\_name) 删除参数

bool has\_param(param\_name) 参数是否存在于参数服务器上

str search\_param() 搜索参数

… … …

时钟相关

函数：

返回值 方法 作用

Time get\_rostime() 获取当前时刻的Time对象

float get\_time() 返回当前时间，单位秒

sleep(duration) 执行挂起

… … …

Time类：

返回值 方法 作用

init(self,secs=0,nsecs=0) 构造函数

Time now() 静态方法 返回当前时刻的Time对象

… … …

Duration类：

返回值 方法 作用

init(self,secs=0,nsecs=0) 构造函数

… … …

topic in rospy

用Python写一个节点间消息收发的demo，创建一个自定义的gps类型的消息，一个节点发布模拟gps信息，另一个接收和计算原点的距离。

1. 自定义消息的生成

string state

float x

float y

2. CMakeLists.txt和package.xml

CMakeLists.txt

find\_package(catkin REQUIRED COMPONENTS

rospy

std\_msgs

message\_generation

)

add\_message\_files(

FILES

gps.msg

)

generate\_messages(DEPENDENCIES std\_msgs)

package.xml

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

<run\_depend>message\_runtion</run\_depend>

对创建的msg进行catkin\_make会在~/catkin\_ws(workspace)/devel/lib/python2.7/dist-packages/topic\_demo下生成msg模块（module），即可在Python程序中通过from test.msg import gps进行gps类型的消息的操作。

3. 消息发布节点

文件位置：test/scripts/pytalker.py

#!/usr/bin/env python

#coding = utf-8

import rospy

from test.msg import gps

def talker():

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

pub = rospy.Publisher("gps\_info",gps,queue\_size=10) #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，会设置成为阻塞式同步收发模式

#queue\_size:0 不建议，会设置为无限缓冲区模式，危险

#queue\_size:10 or more 一般情况下设为10，queue\_size太大会导致数据延时不同步

#更新频率是1hz

rate = rospy.Rate(1)

x = 1.0

y = 2.0

state = "working"

while not rospy.is\_shutdown():

rospy.loginfo('talker:gps:x=%f,y=%f')

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

x = 2\*x

y = 2\*y

rate.sleep()

if \_\_name\_\_ = "\_\_main\_\_":

talker()

与C++代码区别：

rospy创建和初始化一个node，不再需要NodeHandle（rospy没有设计NodeHandle这个句柄），创建topic、service等操作都直接用rospy里对应的方法即可。

rospy节点的初始化补衣服ing必须放在开头，在Publisher建立后再初始化也可以。

消息的创建更加简单，例如gps类型的消息可以直接用类似与构造函数的方式gps(state,x,y)创建。

日志的输出方式不同，C++是ROS\_INFO()，Python中为rospy.loginfo()。

判断节点是否关闭的函数不同，C++为ros::ok()，Python为rospy.is\_shutdown()。

roscpp 和rospy的接口并不一致，实现原理上两套客户端库也有各自的实现，没有基于一个同意的核心库来开发，要避免混用。

ROS2就解决了这一问题，ROS2中的客户端库有rclcpp（ROS Client Library C++）、rclpy（ROS Client Library Python）和其他语言版本，都是基于一个共同的核心ROS客户端库rcl开发（核心库由C语言实现）。

3. 消息订阅节点

文件位置：test/scripts/pylistener.py:

!/usr.bin/env python

#coding = utf-8

import rospy

import math

#导入mgs

from test.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()

在订阅节点中，rospy没有spinOnce()，只有spin()。

# 参考文献

<https://blog.csdn.net/u013832707/article/details/53980759>

<https://blog.csdn.net/weixin_44682965/article/details/107748350>

<http://wiki.ros.org/rospy/Tutorials>

ROS学习笔记rospy介绍：https://blog.csdn.net/weixin\_44682965/article/details/107748350

<https://community.bwbot.org/topic/499>