山东大学 计算机科学与技术 学院

课程实验报告

学号：202100130052	姓名：刘欣月	班级：人工智能班
实验题目：机器人操作系统 ROS 自定义消息的发布和订阅		
实验学时：2	实验日期：20210510	
实验目的：机器人操作系统 ROS 应用，学习发布者设计和订阅者设计，熟悉自定义消息，编写代码完成消息的发布和订阅。		
实验环境：Ubuntu１６　ＲＯＳ		

实验步骤：

实验五：自定义消息的发布和订阅

1. 发布者（Publisher）设计

（1）创建功能包

使用命令 cd ~/catkin_ws/src

Catkin)create_pkg learning_topic roscpp rospy std_msgs geometrt_msgs turtlesim 创建功能包 learning_topic 时使用了依赖 roscpp,rospy,std_msgs,geometry_,sgs,turtlesim 等功能包，

```
liuxinyue@ubuntu:~$ cd ~/catkin_ws/src
liuxinyue@ubuntu:~/catkin_ws/src$ catkin_create_pkg learning_topic roscpp ros
py std_msgs geometry_msgs turtlesim
Created file learning_topic/package.xml
Created file learning_topic/CMakeLists.txt
Created folder learning_topic/include/learning_topic
Created folder learning_topic/src
Successfully created files in /home/liuxinyue/catkin_ws/src/learning_topic. P
lease adjust the values in package.xml.
liuxinyue@ubuntu:~/catkin_ws/src$
```

（2）创建发布者 Publisher 程序，进入 src 文件夹建立

velocity_publisher.cpp 文件,然后输入 cpp 代码如下所示：该代码能够让小海龟以 0.5 线速度 0.2 角速度运动。

```cpp
#include <ros/ros.h>
#include <geometry_msgs/Twist.h>

int main(int argc, char **argv)
{
        // ROS节点初始化
        ros::init(argc, argv, "velocity_publisher");

        // 创建节点句柄
        ros::NodeHandle n;

        // 创建一个Publisher，发布名为/turtle1/cmd_vel的topic，消息类型为
geometry_msgs::Twist，队列长度10
        ros::Publisher turtle_vel_pub = n.advertise<geometry_msgs::Twist>("/
turtle1/cmd_vel", 10);

        // 设置循环的频率
        ros::Rate loop_rate(10);

        int count = 0;
        while (ros::ok())
        {
            // 初始化geometry_msgs::Twist类型的消息
                geometry_msgs::Twist vel_msg;
                vel_msg.linear.x = 0.5;
                vel_msg.angular.z = 0.2;

            // 发布消息
                turtle_vel_pub.publish(vel_msg);
                ROS_INFO("Publsh turtle velocity command[%0.2f m/s, %0.2f
rad/s]",
                                vel_msg.linear.x, vel_msg.angular.z);

            // 按照循环频率延时
            loop_rate.sleep();
        }

        return 0;
}
```

（3）修改编译配置文件并编译功能包，修改 CMakeFlie 文件，在合适的地方插入 add_executable(velocity_publisher src/velocity_puublisher.cpp)

target_link_libraries(velocity_publisher ${catkin_LIBRARIES})

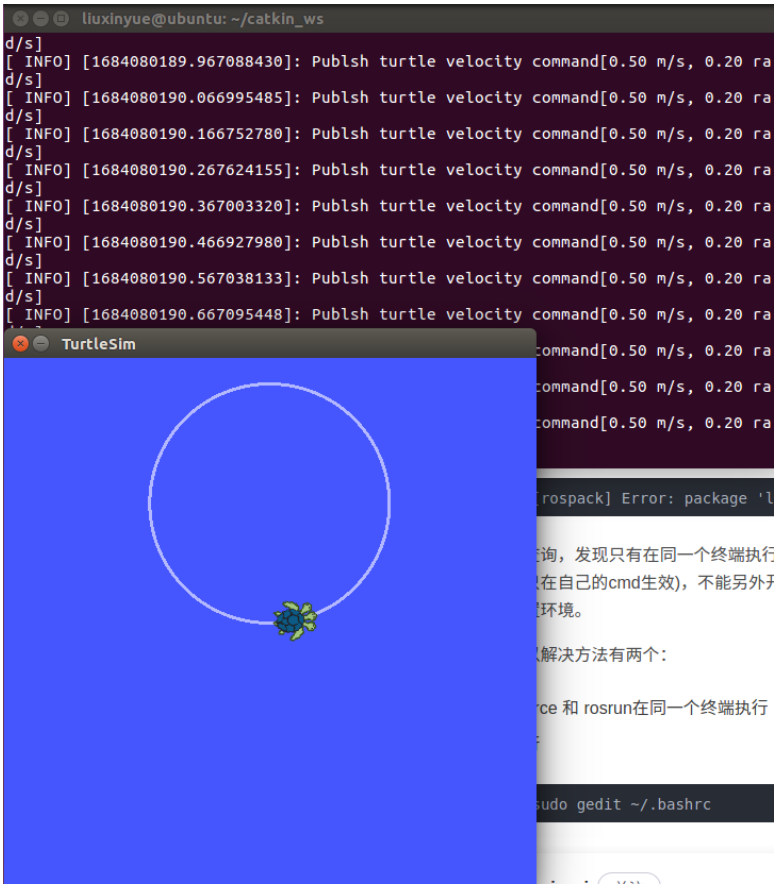
然后执行 catkin_make 语句，结果如下所示

```
add_executable(velocity_publisher src/velocity_publisher.cpp)
target_link_libraries(velocity_publisher ${catkin_LIBRARIES})
```

```
-- Configuring done
-- Generating done
-- Build files have been written to: /home/liuxinyue/catkin_ws/build
####
#### Running command: "make -j2 -l2" in "/home/liuxinyue/catkin_ws/build"
####
Scanning dependencies of target velocity_publisher
[ 50%] Built target my_hello_world_node
[ 75%] Building CXX object learning_topic/CMakeFiles/velocity_publisher.dir/s
rc/velocity_publisher.cpp.o
[100%] Linking CXX executable /home/liuxinyue/catkin_ws/devel/lib/learning_to
pic/velocity_publisher
[100%] Built target velocity_publisher
liuxinyue@ubuntu:~/catkin_ws$
```

（4）执行发布者 Publisher 程序，打开终端输入

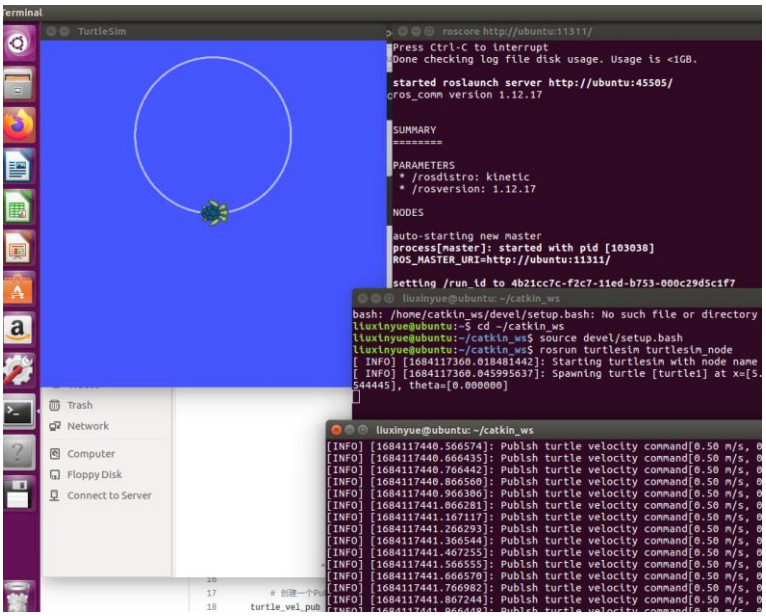
roscore,rosrun turtlesim turtlesim_node rosrun learning_topic velocity_publisher 结果如下所示



（5）编写发布者 Publisher python 程序，在 learning_topic 文件夹中建立 script 文件夹，在 scripts 文件夹中创建 velocity_publihser.py 文件，记得修改文件配置，如下所示：

```python
import rospy
from geometry_msgs.msg import Twist

def velocity_publisher():
        # ROS节点初始化
    rospy.init_node('velocity_publisher', anonymous=True)

        # 创建一个Publisher，发布名为/turtle1/cmd_vel的topic，消息类型为geometry_ms
    turtle_vel_pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)

        #设置循环的频率
    rate = rospy.Rate(10)

    while not rospy.is_shutdown():
                # 初始化geometry_msgs::Twist类型的消息
        vel_msg = Twist()
        vel_msg.linear.x = 0.5
        vel_msg.angular.z = 0.2

                # 发布消息
        turtle_vel_pub.publish(vel_msg)
        rospy.loginfo("Publsh turtle velocity command[%0.2f m/s, %0.2f rad/s]",
                              vel_msg.linear.x, vel_msg.angular.z)

                # 按照循环频率延时
        rate.sleep()

if __name__ == '__main__':
    try:
        velocity_publisher()
    except rospy.ROSInterruptException:
        pass
```

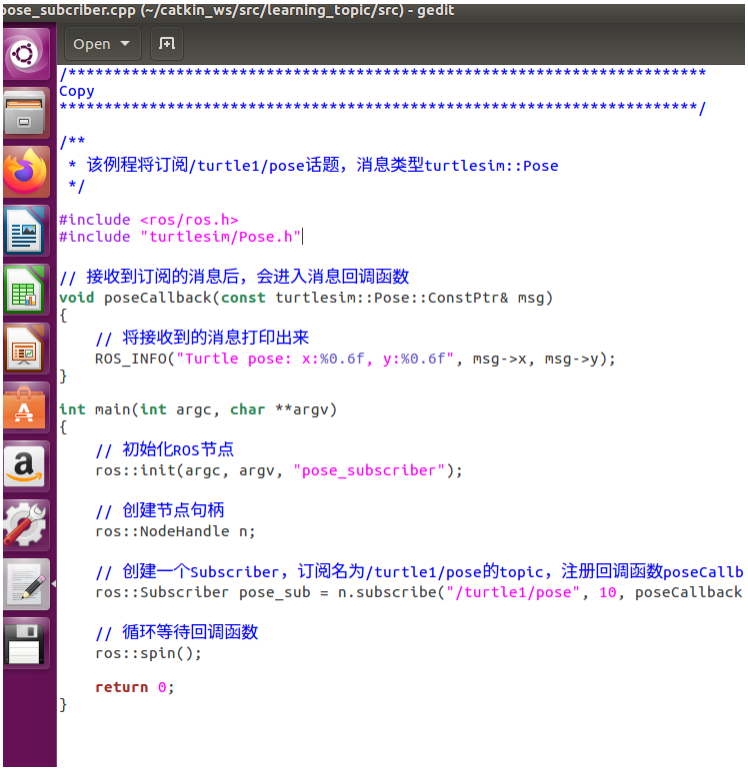
（6）执行发布者 Publisher 程序,在终端输入 roscore,rosrun turtlesim turtlesim_node rosrun learning_topic velocity_publihser.py 得到结果如下所示：



2. 订阅者(subscriber)设计，设计一个消息订阅者来接受 turtlesim 发布的消息

（1）创建订阅者 subscriber 程序，进入 src 文件夹，建立

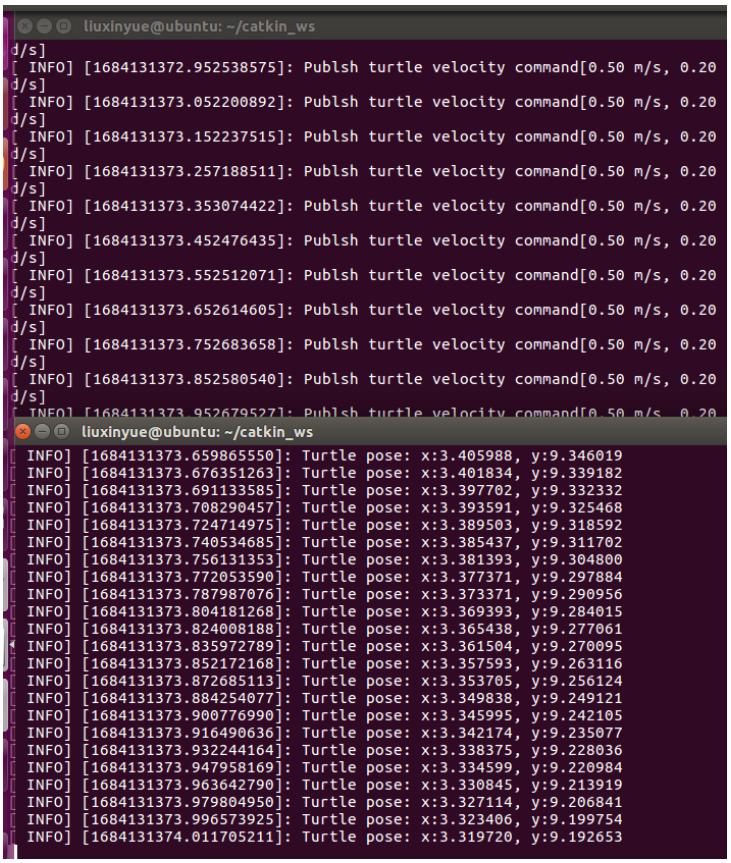
pose_subscriber.cpp 文件，内容如下：



（2）修改编译配置文件并编译功能包，将如下的两个语句插入 CMakeFile.txt 文件，如下所示，然后运行 carkin_make

```
Open
## Rename C++ executable without prefix
## The above recommended prefix causes long target names, the following rename
## target back to the shorter version for ease of user use
## e.g. "rosrun someones_pkg node" instead of "rosrun someones_pkg someones_pk
# set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME node PREFI

## Add cmake target dependencies of the executable
## same as for the library above
# add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )
add_executable(velocity_publisher src/velocity_publisher.cpp)
target_link_libraries(velocity_publisher ${catkin_LIBRARIES})

add_executable(pose_subscriber src/pose_subscriber.cpp)
target_link_libraries(pose_subscriber ${catkin_LIBRARIES})
##############
liuxinyue@ubuntu: ~/catkin_ws
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- ~~  traversing 2 packages in topological order:
-- ~~  - hello_world
-- ~~  - learning_topic
-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-- +++ processing catkin package: 'hello_world'
-- ==> add_subdirectory(hello_world)
-- +++ processing catkin package: 'learning_topic'
-- ==> add_subdirectory(learning_topic)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/liuxinyue/catkin_ws/build
####
#### Running command: "make -j2 -l2" in "/home/liuxinyue/catkin_ws/build"
####
Scanning dependencies of target pose_subscriber
[ 33%] Built target my_hello_world_node
[ 50%] Building CXX object learning_topic/CMakeFiles/pose_subscriber.dir/src/p
e_subscriber.cpp.o
[ 83%] Built target velocity_publisher
[100%] Linking CXX executable /home/liuxinyue/catkin_ws/devel/lib/learning_top
/pose_subscriber
[100%] Built target pose_subscriber
liuxinyue@ubuntu:~/catkin_ws$
```

（3）执行订阅者 subscriber 程序，打开终端输入 roscore rosrun turlesim turtlesim_node tostun learning_topic pose_subscriber，法线 xy 没有变化，运行千米那的发布者节点，再看小海龟运动起来了。

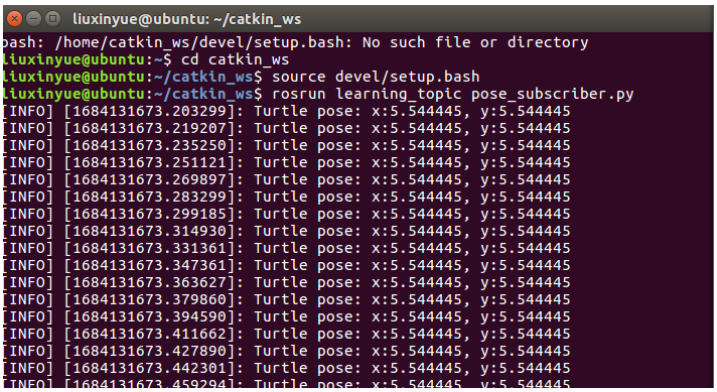


（4）创建订阅者 subscriber 的 python 程序，进入 learning_topic 文件夹下的 scripts 文件夹，建立 pose_subscriber.py 文件，如下所示：

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

#########################################################################
####          Copyright 2020 GuYueHome (www.guyuehome.com).          ###
#########################################################################

# 该例程将订阅/turtle1/pose话题，消息类型turtlesim::Pose

import rospy
from turtlesim.msg import Pose

def poseCallback(msg):
    rospy.loginfo("Turtle pose: x:%0.6f, y:%0.6f", msg.x, msg.y)

def pose_subscriber():
        # ROS节点初始化
    rospy.init_node('pose_subscriber', anonymous=True)

        # 创建一个Subscriber，订阅名为/turtle1/pose的topic，注册回调函数poseCal
    rospy.Subscriber("/turtle1/pose", Pose, poseCallback)

        # 循环等待回调函数
    rospy.spin()

if __name__ == '__main__':
    pose_subscriber()
```

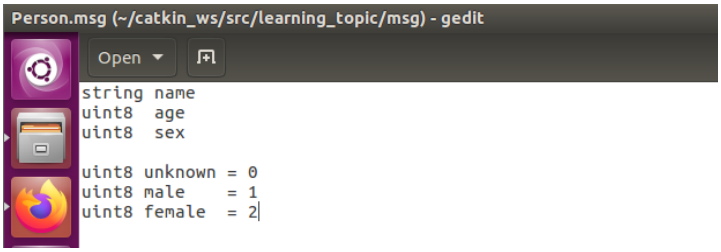
（5）执行订阅者 subscriber 程序，然后在终端输入 roscore

rosrun turtlesim turtlesim_node rosrun learning_topic pose_subscriber.py

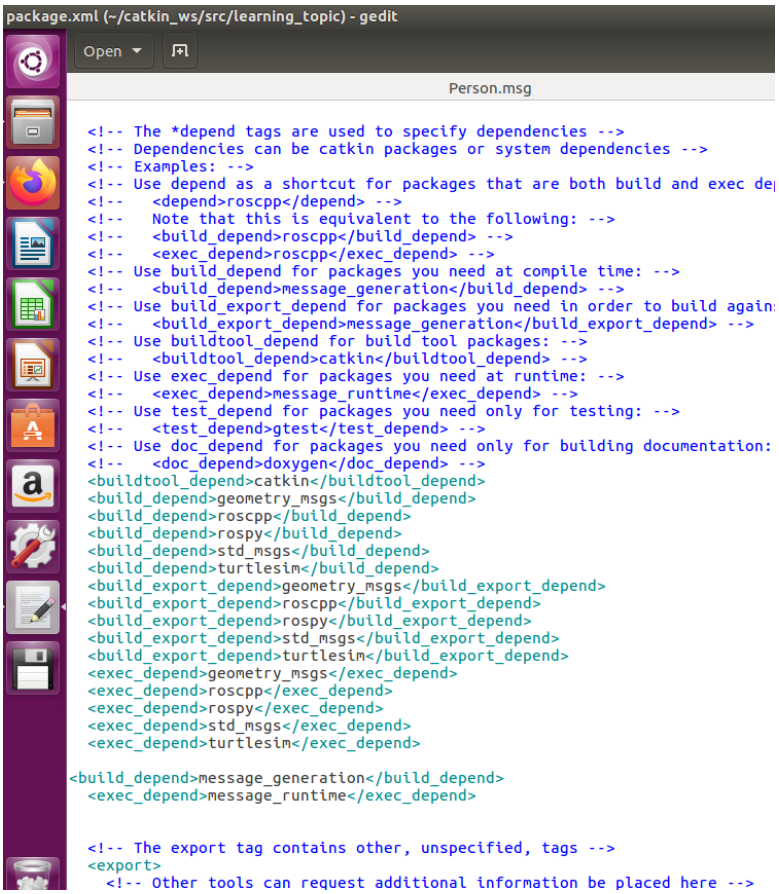


3. 自定义消息类型

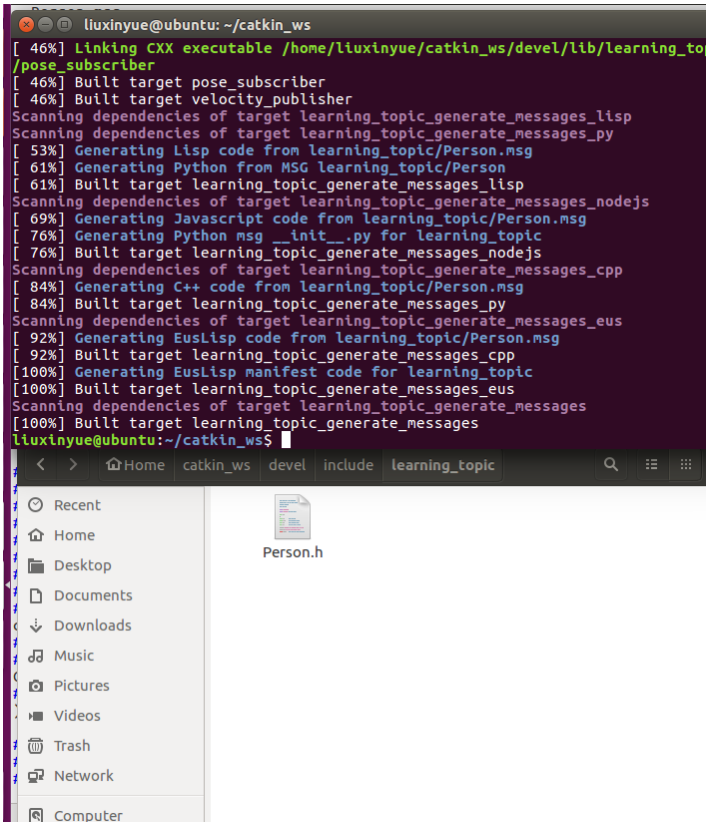
（1）自定义消息类型文件，在 learning_topic 文件夹下建立 msg 文件夹，自定义消息类型如下：



（2）针对自定义消息类型文件添加功能包依赖和编译选项，打开 learning_topic 文件夹下的 package.xml 文件，确保文件中设置了如下的相关依赖：



（3）设计发布者（publisher），执行 catkin_make 命令，通过编译结果如下所示，可以在 catkin_ws 文件夹下 devel 文件夹下 include 文件夹下看到结果：

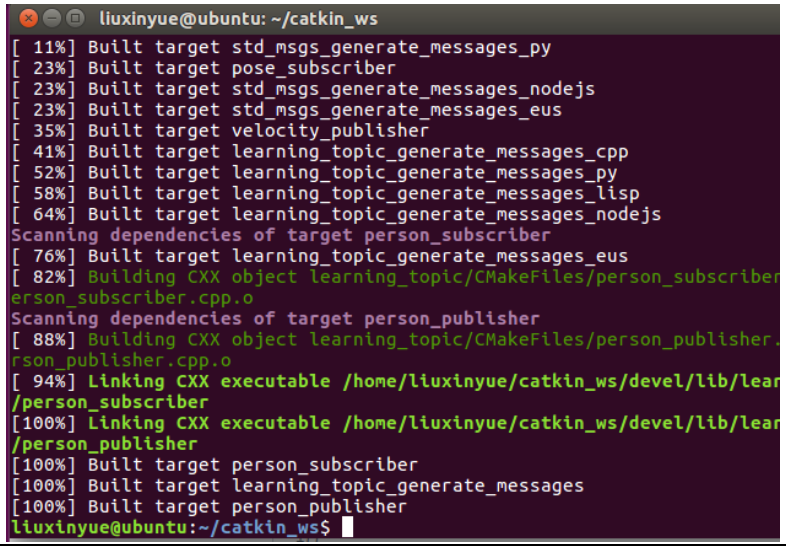


（4）设计订阅者（subscriber），发布名为person_publisher.cpp 内容：

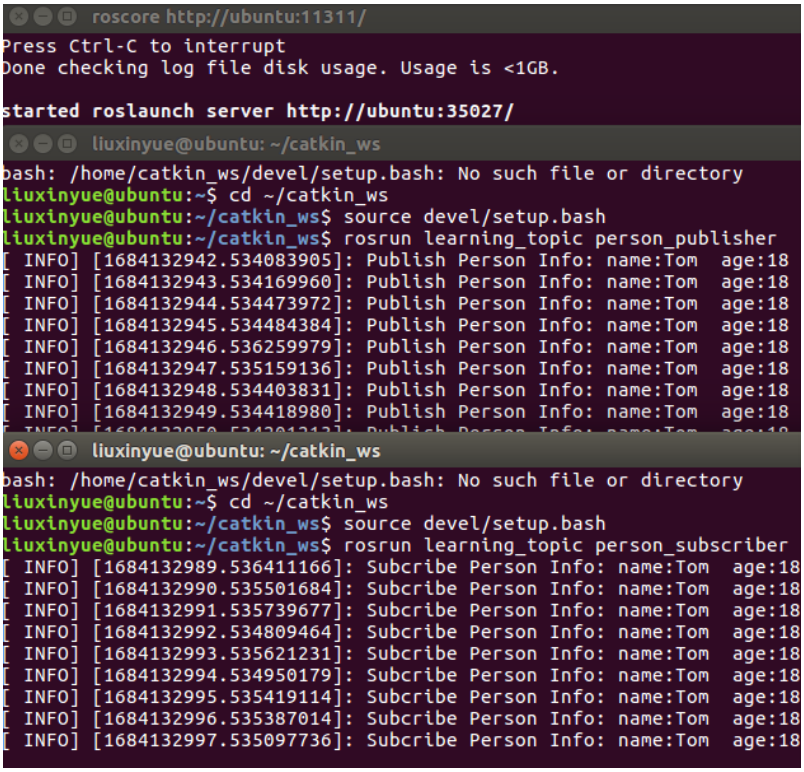
订阅者文件名 person_subscriber.cpp 文件如下所示：

（5）编译发布者（Publisher）和订阅者（subscriber），修改 CMakeFile 文件的配置，插入如下代码；运行 catkin_make

```
add_executable(person_publisher src/person_publisher.cpp)
target_link_libraries(person_publisher ${catkin_LIBRARIES})
add_dependencies(person_publisher ${PROJECT_NAME}_generate_messages_cpp)

add_executable(person_subscriber src/person_subscriber.cpp)
target_link_libraries(person_subscriber ${catkin_LIBRARIES})
add_dependencies(person_subscriber ${PROJECT_NAME}_generate_messages_cpp)
```



（6）执行发布者（Publisher）和订阅者（subscriber），输入 roscore，rosrun learning_topic person_publisher，rosrun learning_topic person_subscriver 结果如下



然后使用 Python 来实现，编写 person_publisher.py 文件和 person_subscriber.py 文件，如下所示：

person_publisher.py (~/catkin_ws/src/learning_topic/scripts) - gedit

Open

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

##########################################################################
####          Copyright 2020 GuYueHome (www.guyuehome.com).          ##
##########################################################################

# 该例程将发布/person_info话题，自定义消息类型learning_topic::Person

import rospy
from learning_topic.msg import Person

def velocity_publisher():
        # ROS节点初始化
    rospy.init_node('person_publisher', anonymous=True)

        # 创建一个Publisher，发布名为/person_info的topic，消息类型为lear
    person_info_pub = rospy.Publisher('/person_info', Person, queue_si

        #设置循环的频率
    rate = rospy.Rate(10)

    while not rospy.is_shutdown():
                # 初始化learning_topic::Person类型的消息
        person_msg = Person()
        person_msg.name = "Tom";
        person_msg.age  = 18;
        person_msg.sex  = Person.male;

                # 发布消息
        person_info_pub.publish(person_msg)
        rospy.loginfo("Publsh person message[%s, %d, %d]",
                                person_msg.name, person_msg.age, perso

                # 按照循环频率延时
        rate.sleep()

if __name__ == '__main__':
    try:
        velocity_publisher()
    except rospy.ROSInterruptException:
        pass
```

person_subscriber.py (~/catkin_ws/src/learning_topic/scripts) - gedit

Open

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

##########################################################################
####          Copyright 2020 GuYueHome (www.guyuehome.com).          ##
##########################################################################

# 该例程将订阅/person_info话题，自定义消息类型learning_topic::Person

import rospy
from learning_topic.msg import Person

def personInfoCallback(msg):
    rospy.loginfo("Subcribe Person Info: name:%s  age:%d  sex:%d",
                  msg.name, msg.age, msg.sex)

def person_subscriber():
        # ROS节点初始化
    rospy.init_node('person_subscriber', anonymous=True)

        # 创建一个Subscriber，订阅名为/person_info的topic，注册回调函数p
    rospy.Subscriber("/person_info", Person, personInfoCallback)

        # 循环等待回调函数
    rospy.spin()

if __name__ == '__main__':
    person_subscriber()
```

之后输入 roscore ，rosrun learning_topic person_publisher.py ，rosrun learning_topic person_subscriber.py