3. ROS2 topic newsletter

Topic communication is one of the most frequently used communication methods in ROS2. Topic communication is based on the publish-subscribe model, where there is a publisher who publishes data on a specified topic, and a subscriber who subscribes to the topic can receive the data as long as it is subscribed to the topic. The next step is to explain how to use Python language to realize topic communication between nodes.

1. New workspace

Terminal input.

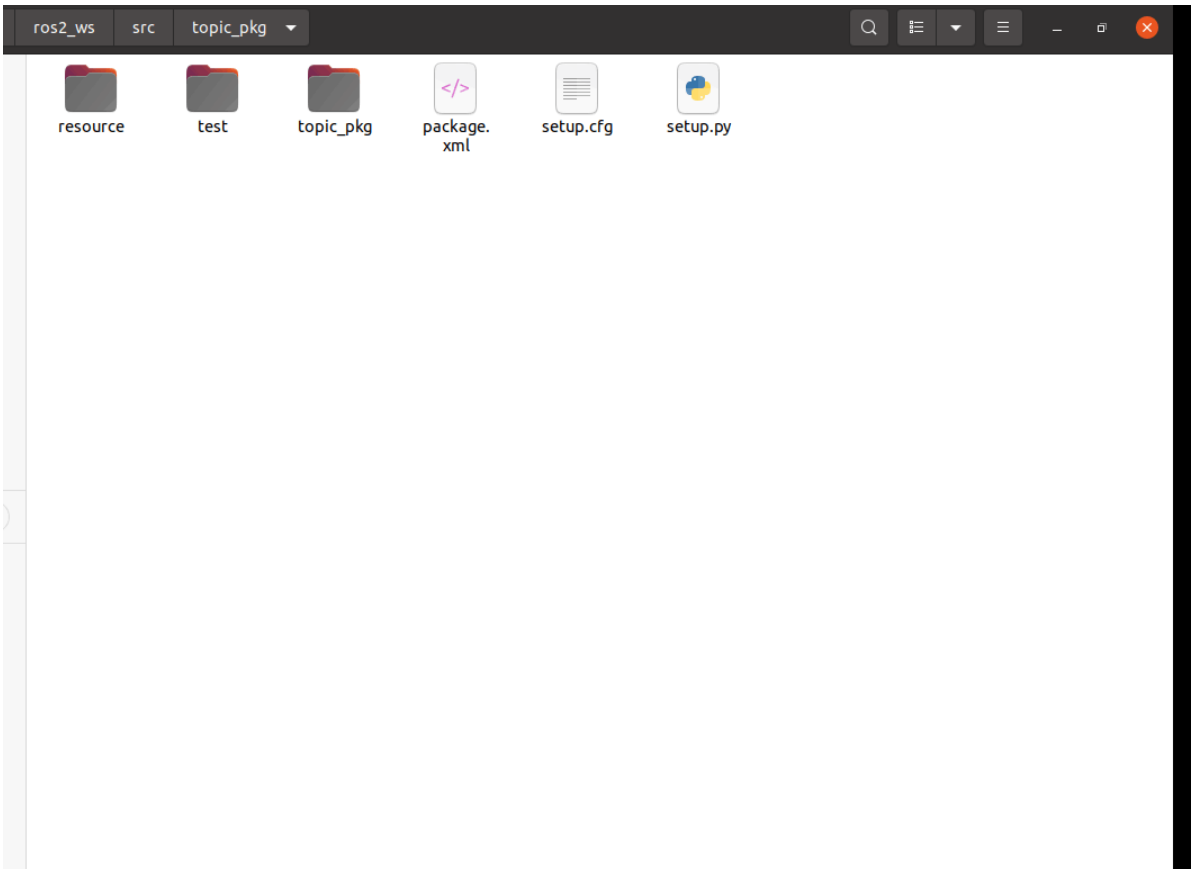
```
mkdir -p ros2_ws/src
cd ros2_ws/src
```

ros2_ws is the name of the workspace and src is the directory where the feature package is stored.

2. New function package

Terminal input,

```
cd ~/ros2_ws/src
ros2 pkg create --build-type ament_python topic_pkg
```



Switching to the topic_pkg folder, there are several files and folders, and the python program we wrote is placed under the topic_pkg folder in that directory,

```
~/ros2_ws/src/topic_pkg/topic_pkg
```

3. Write publisher python file

3.1, Program source code

Create a new file named publisher_demo.py.

```
cd ~/ros2_ws/src/topic_pkg/topic_pkg
gedit publisher_demo.py
```

Copy the following section into the file.

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String
class Topic_Pub(Node):
    def __init__(self,name):
        super().__init__(name)
        self.pub = self.create_publisher(String,"/topic_demo",1)
        self.timer = self.create_timer(1,self.pub_msg)

    def pub_msg(self):
        msg = String()
        msg.data = "Hi,I send a message."
        self.pub.publish(msg)

def main():
    rclpy.init()
    pub_demo = Topic_Pub("publisher_node")
    rclpy.spin(pub_demo)
```

Here is the idea of modularization is used to implement, this approach is conducive to our porting and modifying the code. First of all, a class is defined, the class name is Topic_Pub, there are two parts, one is init, the other is pub_msg. init is to initialize the class itself, pub_msg is the method of the class, that is to say, as long as we create the object of this class, we can use the variables and methods in it.

```
#导入rclpy库 # Import the rclpy library
import rclpy
from rclpy.node import Node
#导入String字符串消息 # Import String messages
from std_msgs.msg import String
#创建一个继承于Node基类的Topic_Pub节点子类 传入一个参数name
# Create a subclass of Topic_Pub node that inherits from the Node base class Pass
in a parameter name
def __init__(self,name):
        super().__init__(name)
        #创建一个发布者，使用create_publisher的函数，传入的参数分别是：
        #话题数据类型、话题名称、保存消息的队列长度
          #Create a publisher, using the function create_publisher, the
parameters passed in are:
        #Topic data type, topic name, length of the queue to save messages in
        self.pub = self.create_publisher(String,"/topic_demo",1)
        #创建一个定时器，间隔1s进入中断处理函数，传入的参数分别是：
```

```
        #中断函数执行的间隔时间，中断处理函数
         # Create a timer that enters the interrupt handler function at 1s
intervals, the parameters passed in are:
        #interrupt function execution interval, interrupt handling function
        self.timer = self.create_timer(1,self.pub_msg)
#定义中断处理函数 # Define interrupt handler functions
def pub_msg(self):
        msg = String()  #创建一个String类型的变量msg
        # Create a variable msg of type String
        msg.data = "Hi,I send a message." #给msg里边的data赋值  # Assign a value to
the data in the msg.
        self.pub.publish(msg) #发布话题数据           #Publishing topic data

#主函数 # main function
def main():
    rclpy.init() #初始化 # Initialization
    pub_demo = Topic_Pub("publisher_node") #创建Topic_Pub类对象，传入的参数就是节点的
名字
     # Create an object of class Topic_Pub, the incoming parameter is the name of
the node
    rclpy.spin(pub_demo) #执行rclpy.spin函数，里边传入一个参数，参数是刚才创建好的
Topic_Pub类对象
    # Execute the rclpy.spin function, passing in a parameter to the Topic_Pub
class object you just created.
```

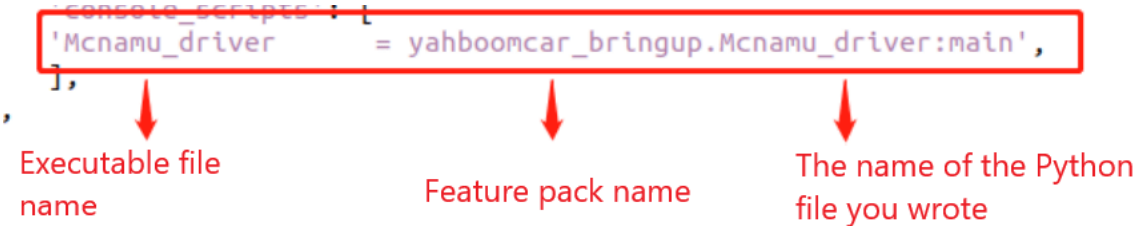
3.2 Modifying the setup.py file

Terminal input.

```
cd ~/ros2_ws/src/topic_pkg
gedit setup.py
```

Find the location shown below.

Inside 'console_scripts': [] add the following, in the format of



The name of the executable file is customized by us, that is, which program we need to execute when we ros2 run, the name of the function package is the function package where your python file is located, followed by the name of the python file, and the main at the end is the entry point of the program execution. As long as the program is written in python format and compiled to generate an executable file, you need to add it here according to the above format.

3.3 Compilation Workspace

```
cd ~/ros2_ws
colcon build
```

The compilation was successful. Add the workspace to the system environment variable by entering the following command.

```
echo "source ~/ros2_ws/install/setup.bash" >> ~/.bashrc
```

Then, reopen the terminal or refresh the environment variables to take effect.

```
source ./.bashrc
```

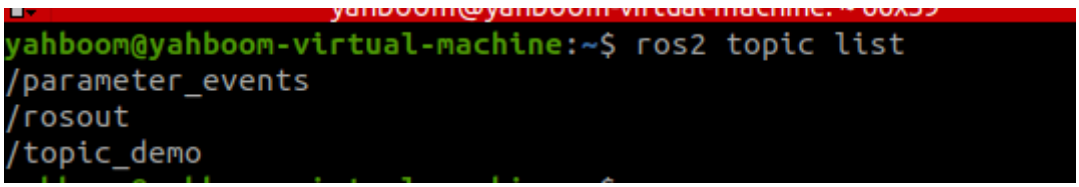
3.4. Run the program

Terminal input,

```
ros2 run topic_pkg publisher_demo
```

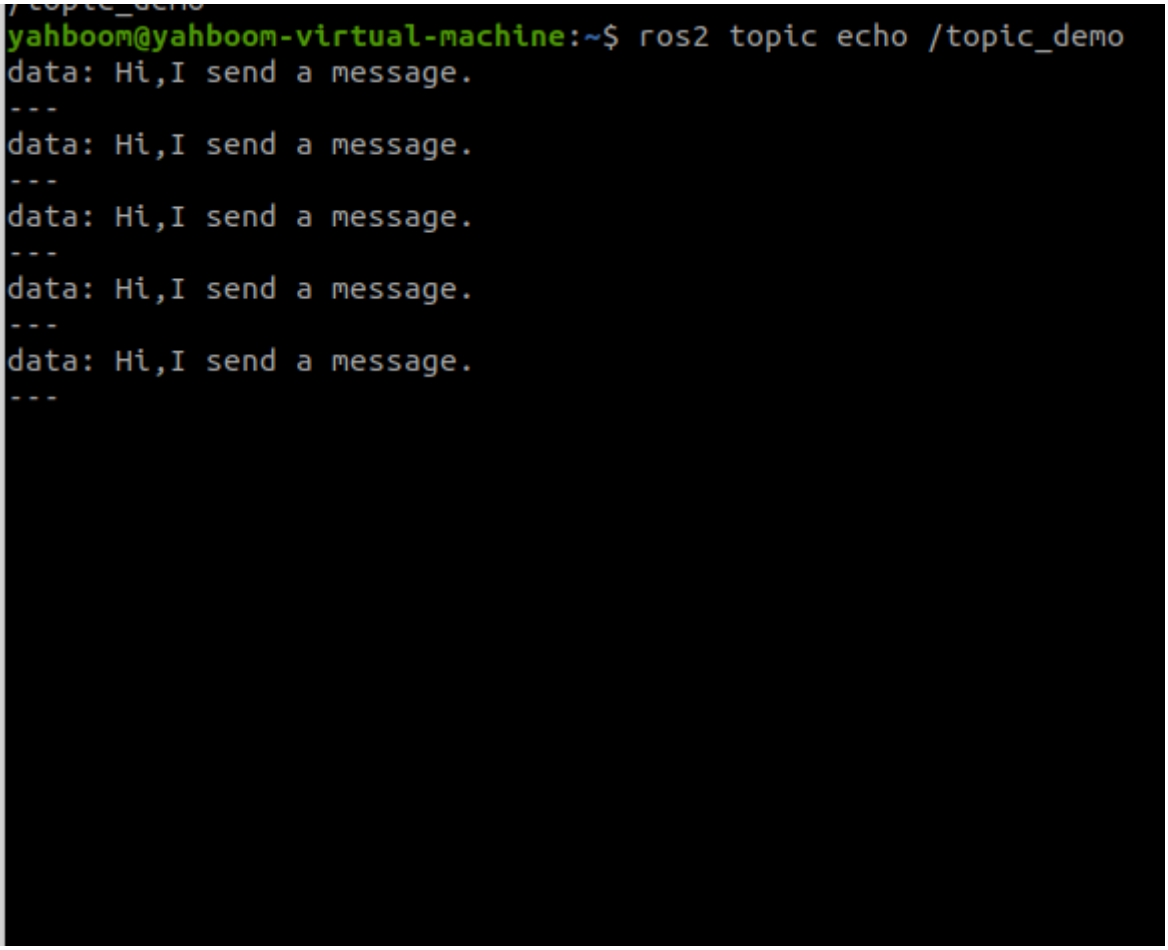
The program is not printing anything after successfully running, we can view the data through the ros2 topic tool, first of all, first of all, check if this has a topic published, terminal input,

```
ros2 topic list
```



This topic_demo is the topic data defined in the program, next, we use ros2 topic echo to print this data, terminal input.

```
ros2 topic echo /topic_demo
```



As you can see, the "Hi,I send a message." printed in the terminal matches the msg.data = "Hi,I send a message." in our code.

4. Write subscriber python file

4.1, Program source code

Create a new file named subscriber_demo.py.

```
cd ~/ros2_ws/src/topic_pkg/topic_pkg
gedit subscriber_demo.py
```

Copy the following section into the file.

```
#导入相关的库 #Import related libraries
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class Topic_Sub(Node):
    def __init__(self,name):
        super().__init__(name)
        #创建订阅者使用的是create_subscription，传入的参数分别是：话题数据类型，话题名称，
回调函数名称，队列长度
        # Create a subscriber using create_subscription with the following
parameters: topic datatype, topic name, callback function name, and queue length
        self.sub =
self.create_subscription(String,"/topic_demo",self.sub_callback,1)
    #回调函数执行程序：打印接收的到信息
    # Callback function executes the program: prints the incoming message.
```

```
    def sub_callback(self,msg):
        print(msg.data)
def main():
    rclpy.init() #ROS2 Python接口初始化
    #ROS2 Python interface initialization
    sub_demo = Topic_Sub("subscriber_node") # 创建对象并进行初始化
    # Create and initialize objects
    rclpy.spin(sub_demo)
```

4.2 Modifying the setup.py file

Terminal input.

```
cd ~/ros2_ws/src/topic_pkg
gedit setup.py
```

Find the location shown below.

```
from setuptools import setup

package_name = 'topic_pkg'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='yahboom',
    maintainer_email='yahboom@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
        'publisher_demo = topic_pkg.publisher_demo:main',
        'subscriber_demo = topic_pkg.subscriber_demo:main'
        ],
    },
)
```

Inside 'console_scripts': [] add the following, in the format of

```
'subscriber_demo = topic_pkg.subscriber_demo:main'
```

'subscriber_demo = topic_pkg.subscriber_demo:main'

4.3. Compilation workspace

```
cd ~/ros2_ws
colcon build
```

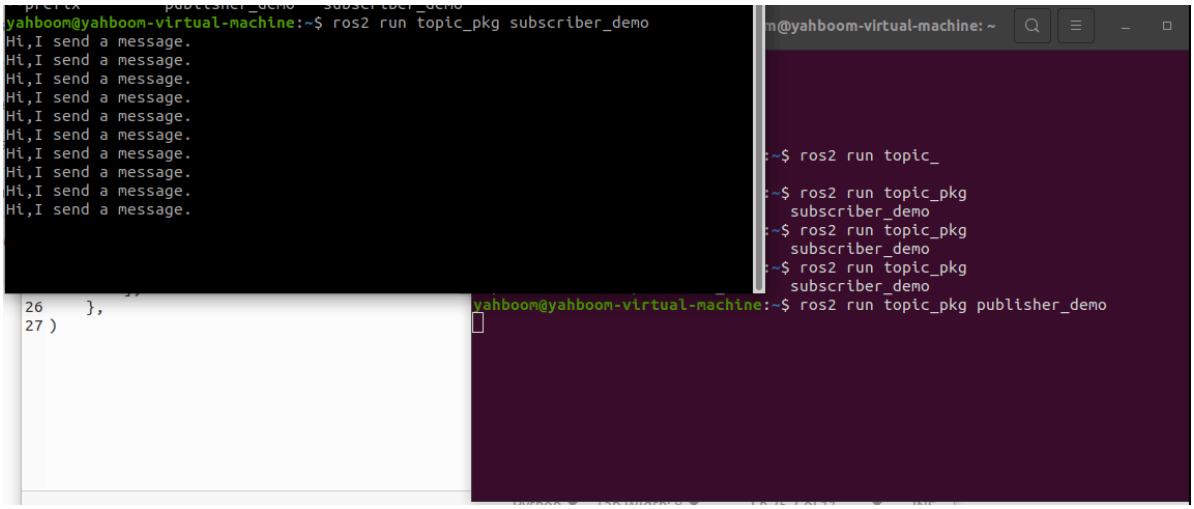
After the compilation is complete, refresh the environment variables in the workspace.

```
source ~/ros2_ws/install/setup.bash
```

4.4. Run the program

Terminal input,

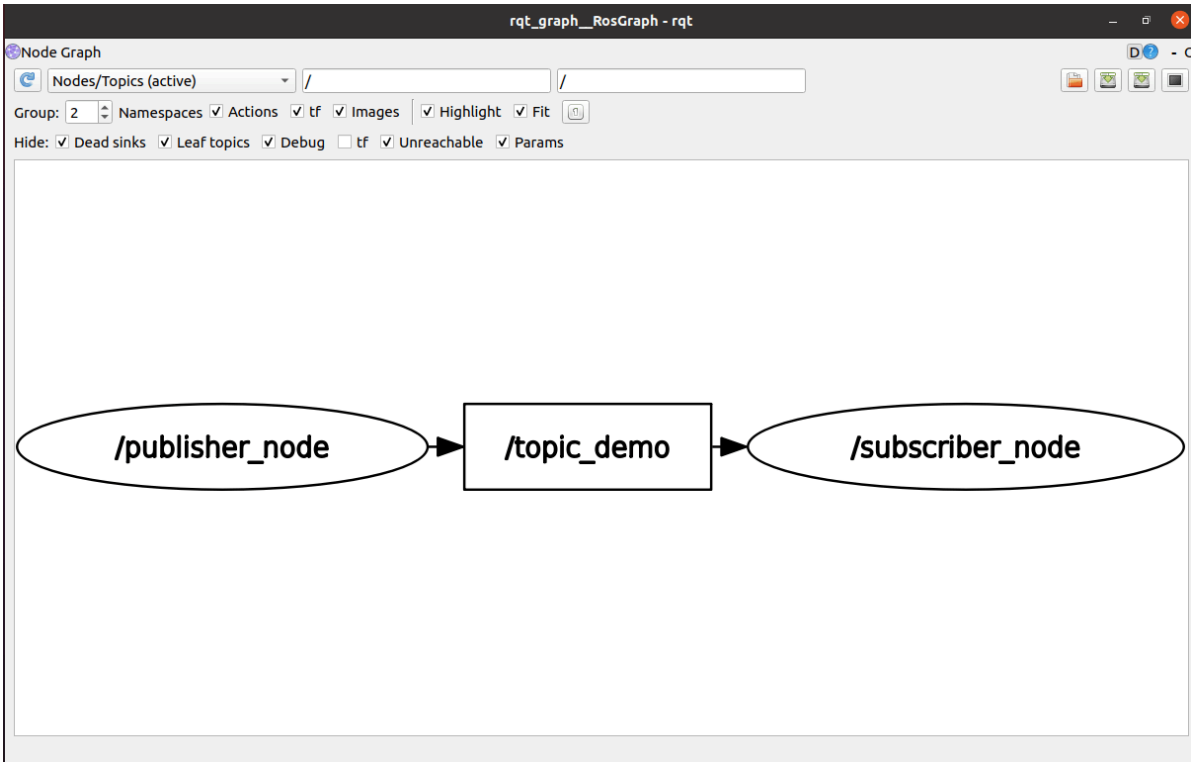
```
#启动发布者节点 # Launch publisher nodes
ros2 run topic_pkg publisher_demo
#启动订阅者节点 #Start subscriber nodes
ros2 run topic_pkg subscriber_demo
```



As shown above, the terminal running the subscriber at this point prints the information about /topic_demo posted by the publisher.

The topic communication between the two nodes can be viewed with the following command.

```
ros2 run rqt_graph rqt_graph
```



5. Summary

5.1, python write topic communication program framework

```
1、导入库文件 Import library file
import rclpy
from rclpy.node import Node
2、创建类 Create class
class ClassName(Node):
    def __init__(self,name):
        super().__init__(name)
        #create subcriber
        self.sub =
self.create_subscription(Msg_Type,Topic_Name,self.CallbackFunction,Msg_Line_Size
)

        #create publisher
        selt.pub = self.create_publisher(Msg_Type,Topic_Name,Msg_Line_Size)

        #create timer
        self.timer = self.create_timer(Timer,self.TimerExcuteFunction)

    def CallbackFunction(self):
        ...
    def TimerExcuteFunction(self):
        ...
3、主函数main:主要是初始化节点，创建对象 The main function: This initializes nodes and
def main():                                creates objects
    rclpy.init()
    class_object = ClassName("node_name")
    rclpy.spin(class_object)
```

5.2. Modify the contents of the setup.py file

To write a node program using python, you need to modify the setup.py file, refer to the above content and add to generate your own node program.