6 Customize topic messages and use

6.1 Custom topic message

Switch to the ~/catkin_ws/src/learning_topic function package directory, and then create a new folder named msg to store custom topic messages.

6.1.1 Define msg file

Switch to the msg directory, create a new blank msg file, and use msg as the suffix to indicate that it is a msg file. Here we take Information.msg as an example to illustrate, and copy the following code into the just created msg file.

```
string  company
string  city
```

6.1.2 Add function package dependencies in package.xml

```
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

6.1.3 Add compile options in CMakeLists.txt

```
Add message_generation to find_package
add_message_files(FILES Information.msg)
generate_messages(DEPENDENCIES std_msgs)
```

6.1.4 Compile and generate language-related files

```
cd ~/catkin_ws
catkin_make
```

6.1.5 C++ language implementation

1. Switch to ~/catkin_ws/src/learning_topic/src, create two new cpp files, name them Information_publisher.cpp and Information_subscriber.cpp, and copy the following codes into them respectively,

Information_publisher.cpp

```
/**
 * This routine will publish the /company_info topic, the message type is a
custom learning_topic::Information
 */

#include <ros/ros.h>
#include "learning_topic/Information.h"

int  main(int  argc, char  ** argv)
{
    // ROS node initialization
    ros::init(argc, argv, "company_Information_publisher");

    // create node handle
    ros::NodeHandle  n;

    // Create a Publisher, publish a topic named /company_info, the message type
is learning_topic::Person, and the queue length is 10
    ros::Publisher  Information_pub  =  n.advertise <
learning_topic::Information >("/company_info", 10);

    // set the frequency of the loop
    ros::Rate  loop_rate(1);

    int  count  =  0;
    while (ros::ok())
{
        // Initialize a message of type learning_topic::Information
        learning_topic::Information  info_msg;
        info_msg.company  =  "Yahboom";
        info_msg.city  =  "Shenzhen";

        // make an announcement
        Information_pub.publish(info_msg);

        ROS_INFO("Information: company:%s  city:%s ",
                  info_msg.company.c_str(), info_msg.city.c_str());


        loop_rate.sleep(); // delay according to loop frequency
}

    return  0;
}
```

Information_subscriber.cpp

```
/**
 * This routine will subscribe to /company_info topic, custom message type
learning_topic::Information
 */

#include <ros/ros.h>
#include "learning_topic/Information.h"
```

```
// After receiving the subscribed message, it will enter the message callback
function to process the data
void  CompanyInfoCallback(const  learning_topic::Information::ConstPtr &  msg)
{
    // print received message
    ROS_INFO("This is: %s  in %s", msg -> company.c_str(), msg -> city.c_str());
}

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

    ros::init(argc, argv, "company_Information_subscriber"); // initialize the
ROS node


    ros::NodeHandle  n; // here is the create node handle

    // Create a Subscriber, subscribe to the topic named topic/company_info, and
register the callback function CompanyInfoCallback
    ros::Subscriber  person_info_sub  =  n.subscribe("/company_info", 10,
CompanyInfoCallback);


    ros::spin(); // loop waiting for callback function

    return  0;
}
```

2. modify the CMakeLists.txt file

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

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

3. the core part

The implementation process here is the same as before, the main difference is the introduction of header files and the use of custom message files:

The import header file is

```
#include "learning_topic/Information.h"
```

The front learning_topic is the function package name, and the back Information.h is the header file name generated by the msg file just created

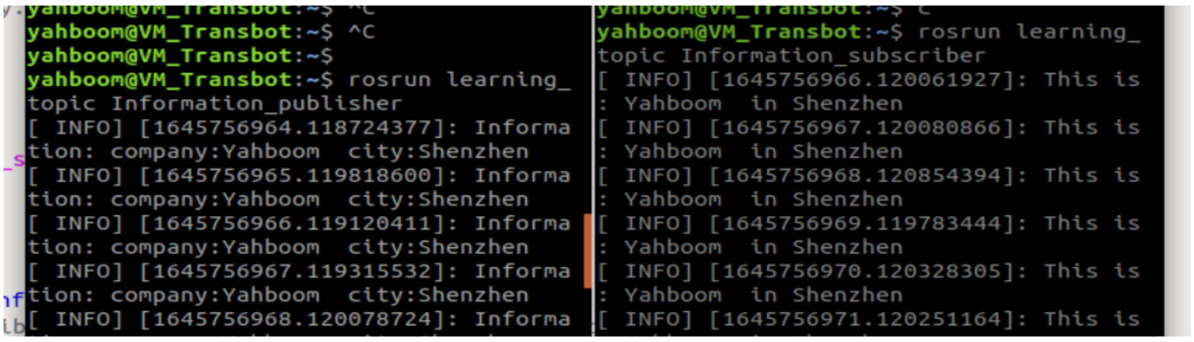
Using a custom message file is

```
learning_topic::Information info_msg;
info_msg.company = "Yahboom";
info_msg.city = "Shenzhen";
void CompanyInfoCallback(const learning_topic::Information::ConstPtr& msg)
```

4), run the program

```
roscore
rosrun learning_topic Information_publisher
rosrun learning_topic Information_subscriber
```

5. run the screenshot



```
yahboom@VM_Transbot:~$ ^C
yahboom@VM_Transbot:~$
yahboom@VM_Transbot:~$ rosrun learning_
topic Information_publisher
[ INFO] [1645756964.118724377]: Informa
tion: company:Yahboom  city:Shenzhen
[ INFO] [1645756965.119818600]: Informa
tion: company:Yahboom  city:Shenzhen
[ INFO] [1645756966.119120411]: Informa
tion: company:Yahboom  city:Shenzhen
[ INFO] [1645756967.119315532]: Informa
tion: company:Yahboom  city:Shenzhen
[ INFO] [1645756968.120078724]: Informa
```

```
yahboom@VM_Transbot:~$ rosrun learning_
topic Information_subscriber
[ INFO] [1645756966.120061927]: This is
: Yahboom  in Shenzhen
[ INFO] [1645756967.120080866]: This is
: Yahboom  in Shenzhen
[ INFO] [1645756968.120854394]: This is
: Yahboom  in Shenzhen
[ INFO] [1645756969.119783444]: This is
: Yahboom  in Shenzhen
[ INFO] [1645756970.120328305]: This is
: Yahboom  in Shenzhen
[ INFO] [1645756971.120251164]: This is
```

6. program description

As a publisher, Information_publisher continuously publishes the content of messages to the topic "/company_info", and prints the published messages; and Information_subscriber, which is a subscriber, also continuously receives the content of the topic "/company_info", and then prints it out in the callback function.

6.1.6 Python language implementation

1. switch to ~/catkin_ws/src/learning_topic/script, create two new py files, named Information_publisher.py and Information_subscriber.py, and copy the following codes into them respectively,

Information_publisher.py

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


import  rospy

from  learning_topic.msg  import  Information  #Import custom msg

 def information_publisher():

    rospy.init_node('information_publisher', anonymous = True) # ROS node
initialization

    # Create a Publisher, publish a topic named /company_info, the message type
is learning_topic::Information, and the queue length is 6
    info_pub  =  rospy.Publisher('/company_info', Information, queue_size = 6)
    rate = rospy.Rate(10) the frequency of the loop

    while  not  rospy.is_shutdown():

    # Initialize messages of type learning_topic::Information
        info_msg  =  Information()
        info_msg.company  =  "Yahboom";
        info_msg.city   =  "Shenzhen";

        info_pub.publish(info_msg) # publish message

        rospy.loginfo("This is %s in %s.", info_msg.company, info_msg.city) #
print the post message

        rate.sleep() # Delay according to the loop frequency

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

Information_subscriber.py

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


import  rospy

from  learning_topic.msg  import  Information  #Import custom msg

 def CompanyInfoCallback(msg):

    rospy.loginfo("company: name:%s city:%s ", msg.company, msg.city) # print
subscription received information

 def Infomation_subscriber():

    rospy.init_node('Infomation_subscriber', anonymous = True) # ROS node
initialization

    # Create a Subscriber, subscribe to a topic named /company_info, and register
the callback function personInfoCallback
    rospy.Subscriber("/company_info", Information, CompanyInfoCallback)

    rospy.spin() # loop waiting for the callback function

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

2. the core part

Here is mainly to explain how to import and use custom message modules:

import

```
from  learning_topic.msg  import  Information
```

use

```
info_msg  =  Person()
info_msg.company  =  "Yahboom";
info_msg.city   =  "Shenzhen";
```

3. run the program

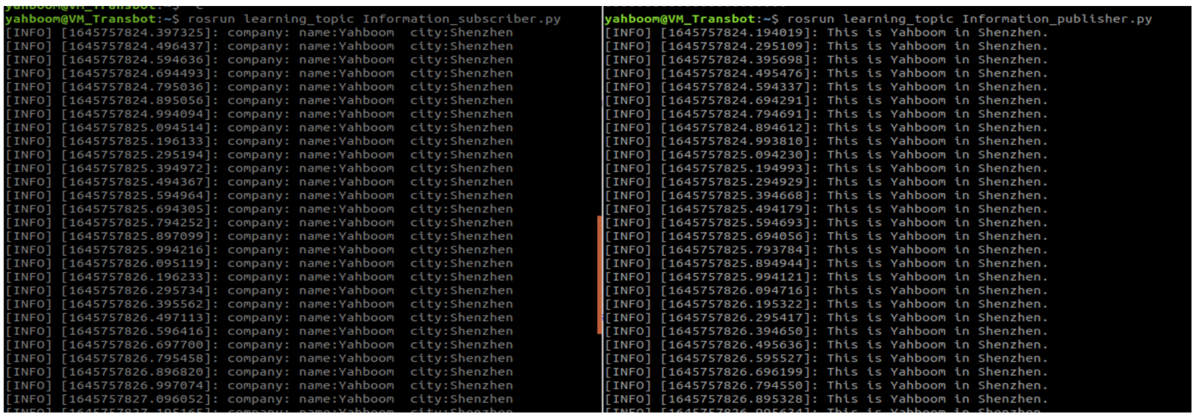
Before running the program, first add executable permissions to the py file

```
sudo chmod a+x Information_subscriber.py
sudo chmod a+x Information_publisher.py
```

run the program

```
roscore
rosrun learning_topic Information_publisher.py
rosrun learning_topic Information_subscriber.py
```

4. run the screenshot



```
yahboom@VM_Transbot:~$ rosrun learning_topic Information_subscriber.py
[INFO] [1645757824.397325]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757824.496437]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757824.594636]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757824.694493]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757824.795036]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757824.895056]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757824.994094]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.094514]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.196133]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.295194]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.394972]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.494367]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.594964]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.694305]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.794252]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.897099]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757825.994216]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.095119]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.196233]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.295734]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.395562]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.497113]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.596416]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.697700]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.795458]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.896820]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757826.997074]: company: name:Yahboom  city:Shenzhen
[INFO] [1645757827.096052]: company: name:Yahboom  city:Shenzhen
```

```
yahboom@VM_Transbot:~$ rosrun learning_topic Information_publisher.py
[INFO] [1645757824.194019]: This is Yahboom in Shenzhen.
[INFO] [1645757824.295109]: This is Yahboom in Shenzhen.
[INFO] [1645757824.395698]: This is Yahboom in Shenzhen.
[INFO] [1645757824.495476]: This is Yahboom in Shenzhen.
[INFO] [1645757824.594337]: This is Yahboom in Shenzhen.
[INFO] [1645757824.694291]: This is Yahboom in Shenzhen.
[INFO] [1645757824.794691]: This is Yahboom in Shenzhen.
[INFO] [1645757824.894612]: This is Yahboom in Shenzhen.
[INFO] [1645757824.993810]: This is Yahboom in Shenzhen.
[INFO] [1645757825.094230]: This is Yahboom in Shenzhen.
[INFO] [1645757825.194993]: This is Yahboom in Shenzhen.
[INFO] [1645757825.294929]: This is Yahboom in Shenzhen.
[INFO] [1645757825.394668]: This is Yahboom in Shenzhen.
[INFO] [1645757825.494179]: This is Yahboom in Shenzhen.
[INFO] [1645757825.594693]: This is Yahboom in Shenzhen.
[INFO] [1645757825.694056]: This is Yahboom in Shenzhen.
[INFO] [1645757825.793784]: This is Yahboom in Shenzhen.
[INFO] [1645757825.894944]: This is Yahboom in Shenzhen.
[INFO] [1645757825.994121]: This is Yahboom in Shenzhen.
[INFO] [1645757826.094716]: This is Yahboom in Shenzhen.
[INFO] [1645757826.195322]: This is Yahboom in Shenzhen.
[INFO] [1645757826.295417]: This is Yahboom in Shenzhen.
[INFO] [1645757826.394650]: This is Yahboom in Shenzhen.
[INFO] [1645757826.495636]: This is Yahboom in Shenzhen.
[INFO] [1645757826.595527]: This is Yahboom in Shenzhen.
[INFO] [1645757826.696199]: This is Yahboom in Shenzhen.
[INFO] [1645757826.794550]: This is Yahboom in Shenzhen.
[INFO] [1645757826.895328]: This is Yahboom in Shenzhen.
```