ros实操

参考

1 建工作空间

```
cd /your_path
mkdir catkin_ws/src
cd catkin_ws
catkin_make # initialize
```

2 创建package

```
catkin_create_pkg topic_exercise roscpp rospy std_msgs
# catkin_create_pkg pkg_name depend_pkg
```

out:

```
Created file topic_exercise/package.xml
Created file topic_exercise/CMakeLists.txt
Created folder topic_exercise/include/topic_exercise
Created folder topic_exercise/src
Successfully created files in /home/mi804/robots/ZJUS/src/topic_exercise.
Please adjust the values in package.xml.
```

主要生成了 package.xml和CMakeLists.txt两个文件，要注意文件结构。

当然也可以不用命令行创建package，自己手动创建文件结构，并创建package.xml和CMakeLists.txt（可以复制其他的package的文件，然后做相应更改）

3.0 roscpp中node，topic，service

一般步骤

1. 调用`ros::init()`函数，从而初始化**节点的名称和其他信息**，一般我们ROS程序一开始都会以这种方式开始。
2. 创建`ros::NodeHandle`对象，也就是节点的句柄，它可以用来创建Publisher、Subscriber以及做其他事情。
3. 开始你的表演

```
#include<ros/ros.h>
int main(int argc, char** argv)
{
    ros::init(argc, argv, "your_node_name");
    ros::NodeHandle nh;
    //....节点功能
    //....
    ros::spin();//用于触发topic、service的响应队列
    return 0;
}
```

```
//创建话题的publisher
ros::Publisher advertise(const string &topic, uint32_t queue_size, bool
latch=false);
//第一个参数为发布话题的名称
//第二个是消息队列的最大长度，如果发布的消息超过这个长度而没有被接收，那么就的消息就会出队。通常设为一个较小的数即可。
//第三个参数是是否锁存。某些话题并不是会以某个频率发布，比如/map这个topic，只有在初次订阅或者地图更新这两种情况下，/map才会发布消息。这里就用到了锁存。

//创建话题的subscriber
ros::Subscriber subscribe(const string &topic, uint32_t queue_size, void(*)
(M));
//第一个参数是订阅话题的名称
//第二个参数是订阅队列的长度，如果受到的消息都没来得及处理，那么新消息入队，就消息就会出队
//第三个参数是回调函数指针，指向回调函数来处理接收到的消息

//创建服务的server，提供服务
ros::ServiceServer advertiseService(const string &service, bool(*srv_func)
(Mreq &, Mres &));
//第一个参数是service名称
//第二个参数是服务函数的指针，指向服务函数。指向的函数应该有两个参数，分别接受请求和响应。

//创建服务的client
ros::ServiceClient serviceClient(const string &service_name, bool
persistent=false);
//第一个函数式service名称
//第二个参数用于设置服务的连接是否持续，如果为true，client将会保持与远程主机的连接，这样后续的请求会快一些。通常我们设为flase
```

3 创建node (以talker和listener为例)

(1) 创建两个文件talker.cpp和listener.cpp，在/topic_exercise/src下

talker.cpp:

```
#include <ros/ros.h>
#include <topic_exercise/gps.h>  //自定义msg产生的头文件

int main(int argc, char **argv)
{
  ros::init(argc, argv, "talker");  //用于解析ROS参数，第三个参数为本节点名
  ros::NodeHandle nh;     //实例化句柄，初始化node

  topic_exercise::gps msg;  //自定义gps消息并初始化

  ros::Publisher pub = nh.advertise<topic_exercise::gps>("gps_info", 1); //创建
publisher，往"gps_info"话题上发布消息
  ros::Rate loop_rate(1.0);    //定义发布的频率，1HZ
  while (ros::ok())    //循环发布msg
  {
    pub.publish(msg);//以1Hz的频率发布msg
    loop_rate.sleep();//根据前面的定义的loop_rate,设置1s的暂停
  }
  return 0;
}
```

listener.cpp

```
#include <ros/ros.h>
#include <topic_exercise/gps.h>
#include <std_msgs/Float32.h>

void gpsCallback(const topic_exercise::gps::ConstPtr &msg)
{
    std_msgs::Float32 distance;  //计算离原点(0,0)的距离
    distance.data = sqrt(pow(msg->x,2)+pow(msg->y,2));
    ROS_INFO("Listener: Distance to origin = %f, state: %s",distance.data,msg->state.c_str()); //输出
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "listenerss");  //这里的listener这个名字会作为node的name出现在rosnode list里
  ros::NodeHandle n;  //node句柄
  ros::Subscriber sub = n.subscribe("gps_info", 1, gpsCallback);  //设置回调函数gpsCallback
  ros::spin(); //ros::spin()用于调用所有可触发的回调函数，将进入循环，不会返回，类似于在循环里反复调用spinOnce()
  //而ros::spinOnce()只会去触发一次
  return 0;
}
```

在 Cmakelist.txt中需要添加

```
add_executable(talker src/talker.cpp)
add_dependencies(talker topic_exercise_generate_messages_cpp)
#表明在编译talker前，必须先生编译完成自定义消息
#必须添加add_dependencies，否则找不到自定义的msg产生的头文件
#表明在编译talker前，必须先生编译完成自定义消息，自定义消息的部分在后面
target_link_libraries(talker ${catkin_LIBRARIES}) #链接

add_executable(listener src/listener.cpp)
add_dependencies(listener topic_exercise_generate_messages_cpp)
target_link_libraries(listener ${catkin_LIBRARIES})#链接
```

(3) 自定义msg

在topic_exercise/msg下新建gps.msg

```
string state   #工作状态
float32 x      #x坐标
float32 y      #y坐标
```

在CmakeList.txt修改

```
find_package(catkin REQUIRED COMPONENTS
roscpp
std_msgs
message_generation    #必须添加
)

add_message_files(
```

```
  FILES
  gps.msg  # 自定义的文件，不用添加路径/msg
)
#catkin在cmake之上新增的命令，指定从哪个消息文件生成

generate_messages(
  DEPENDENCIES
  std_msgs
)
#catkin新增的命令，用于生成消息
#DEPENDENCIES后面指定生成msg需要依赖其他什么消息，由于gps.msg用到了flaot32这种ROS标准消息，因此需要再把std_msgs作为依赖
```

package.xml需要添加：

```
<build_depend>message_generation</build_depend>
<run_depend>message_runtime</run_depend>
```

msg等同于各个节点都知道的结构体，自定义msg即让编译器帮忙生成了这样的结构体，声称在了工作空间的devel下。可以看做：

```
struct gps
{
    string state;
    float32 x;
    float32 y;
}
```

4 运行listener和talker

4.1 node方式

每开一个终端需要

```
source devel/setup.bash
```

以下每个代码框代表新开一个终端

```
roscore #启动master
```

```
rosrun topic_exercise talker
```

```
rosrun topic_exercise listener
```

以下为使用ros的终端命令示例：

```
rosnode list  #节点信息
rostopic list   #topic信息
rostopic echo /gps_info #topic内容
```

更多终端命令可以查看教程或者百度

4.2 launch 方式

在launch/下新建topic_exercise.launch

```
<launch>
    <node pkg="topic_exercise" type="talker" name="talker_in_launch"
output="screen">
    </node>
    <node pkg="topic_exercise" type="listener" name="listener_in_launch"
output="screen">
    </node>
</launch>
```

注意：上面的type中必须与Cmakelists.txt中添加的target一样，即下面这几句：

```
add_executable(listener src/listener.cpp ) #生成可执行文件listener
add_dependencies(listener topic_demo_generate_messages_cpp)
target_link_libraries(listener ${catkin_LIBRARIES})#链接
```

launch文件中还可以添加很多其他的东西，比如param参数。

5 service实操

(1)自定义srv

(2)定义server和client，修改cmake

(3)运行，先跑server，再跑client

(4)可选，ros内写socket server，在另外的程序（非ros）（如直接写python程序写socket client），两者进行通讯。也可以在另外的电脑上跑程序（两者要在同一局域网）。当然，也可以用c++写

简单的socket程序：

server.py

```
import socket
phone=socket.socket(socket.AF_INET,socket.SOCK_STREAM) #买手机
phone.bind(('127.0.0.1',8080)) #server ip

phone.listen(5) #开机，backlog

print('starting....')
conn,addr=phone.accept() #接受client
print(conn)
print('client addr',addr)
print('ready to read msg')
client_msg=conn.recv(1024) #收消息
print('client msg: %s' %client_msg)
conn.send(client_msg.upper()) #发消息

conn.close()
phone.close()
```

client.py

```
import socket
phone=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
phone.connect(('127.0.0.1',8080)) #server ip

phone.send('hello'.encode('utf-8')) #发消息

back_msg=phone.recv(1024)
print(back_msg)

phone.close()
```

6 param实操

写一个node，需要使用参数服务器（api调用方式获取参数），然后在launch文件中启动节点，并给参数（可以尝试直接给参数和利用参数文件的方式）