话题 通信 妻子一个发布在话题一个信息 ，需要这个信息 从相应的话题 拿到信息

这就算话题通信 发布订阅模式

服务通信 是 类似我访问一个网页 之后这个网页把相应的信息给你

参赛服务器 有一个容器 里面存储信息， 这里面的信息是共享的

不同节点 都可以向这个容器 写入和读取数据

掌握各种通信机制

之后每种通信机制的理论模型

和代码实现

一些ros的基础操作

写几个例子

* 能够熟练介绍ROS中常用的通信机制
* 能够理解ROS中每种通信机制的理论模型
* 能够以代码的方式实现各种通信机制对应的案例
* 能够熟练使用ROS中的一些操作命令
* 能够独立完成相关实操案例

话题通信的场景

发布订阅模式

一个节点发布消息 一个节点订阅消息 通过节点耦合

机器人在执行导航功能，使用的传感器是激光雷达，机器人会采集激光雷达感知到的信息并计算，然后生成运动控制信息驱动机器人底盘运动。

在上述场景中，就不止一次使用到了话题通信。

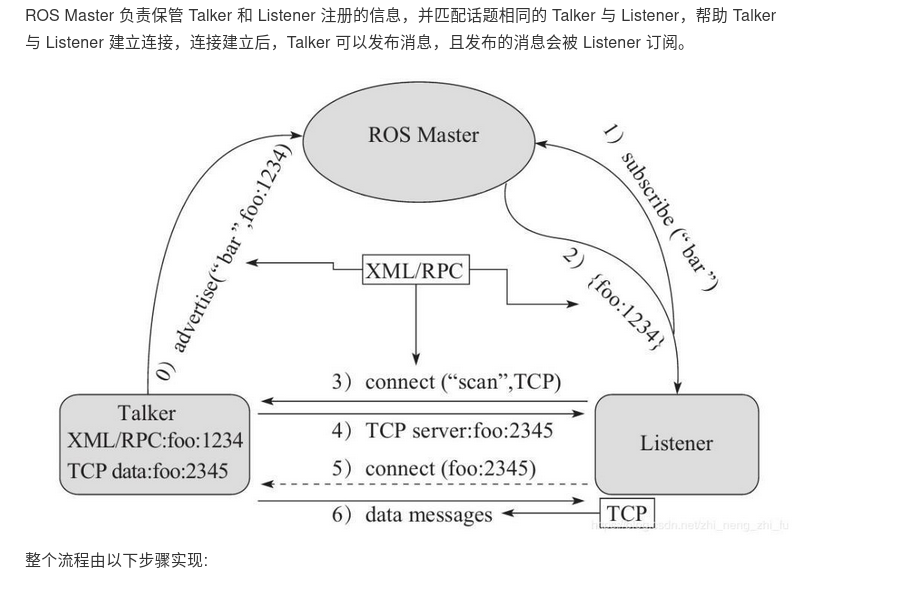
* 以激光雷达信息的采集处理为例，在 ROS 中有一个节点需要时时的**发布当前雷达采集到的数据，导航模块中也有节点会订阅**并解析雷达数据。
* 再以运动消息的发布为例，**导航模块会根据传感器采集的数据时时的计算出运动控制信息**并发布给底盘，**底盘也可以有一个节点订阅运动信息**并最终转换成控制电机的脉冲信号。

话题通信适用于 数据不停更新的摸索

通过发布 订阅 实现节点间的通信 一方发布信息 一方订阅信息

不断更新少逻辑处理的场景

话题通信的理论模型

角色：

ROS Master ： 管理者

Talker：发布信息

Listener：订阅Talker的信息

实现流程：

master 让 talker 和 listener建立一个连接 ，让2者可以通信

第0 talker在maseter 注册 自己的话题 自己的RPC地址

第1 listener 在maseter主车 自己订阅的话题

第2 mastet 比对发布者 订阅者的话题进行比对， 如果话题一致 会把发布者的地质 给lisetener

第3listener 通过RPC地址与talker建立连接 访问talker

第4 talker会响应 会包含TCP地址 把这些信息 给listener

第5 listener通过 TCP地址 与listener建立连接， listener访问了talker（根据TCP地址）

第6 talker通过TCP地址 向listener向talker发送信息

注意事项：

使用的协议是包含RCP 和 TCP

0-4是RCP 建立连接

5-6是TCP传递信息

0 和1 两步没有先后关系

发布方和订阅方的数量，发布方和订阅方 可以存在多个

多个talker和多个listener 可以同时存在多个

talker和listener建立连接后，master可以关闭

这些流程在ros已经封装，在后续可以直接调用

掉包：

编写 talker和listener

设置话题

关注消息的发布

实现的关键点

发布方talker

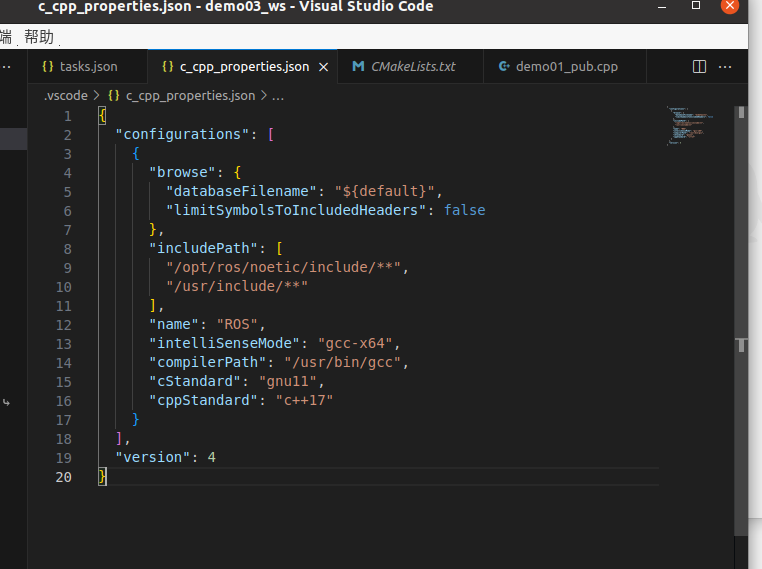
接收方 listener

数据

发布方 talker

这两个是设置内容 让写vscode的文件的时候出现提示 一个是在task加入group 那一行的内容

一个是改称c++17



首先

ros::Nodehandle nh; 创建个节点句柄

ros::Publisher 是创建一个发布者 nh.advertise() 出现的那个提示



有多个重载 这个返回值是ros::Publisher

之后<M> 数据的发布类型

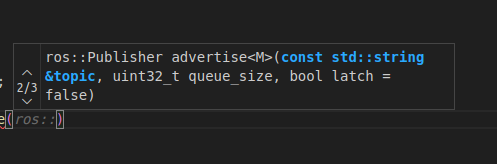
const std::string &topic 传入一个不可变的实参 这个是话题的名字

uint\_32发布的数据长度 queue size 发布的数据如果很多 部分数据每发出去，这个就算我们可以保留的数据长度

例子 假设我们设置为 queue\_size=10 我们发送12条数据 如果堵塞，一条都没有发出去

12条数据 就会放到一个队列里，因为我们设置的是10 所以最开始的两条被舍弃了

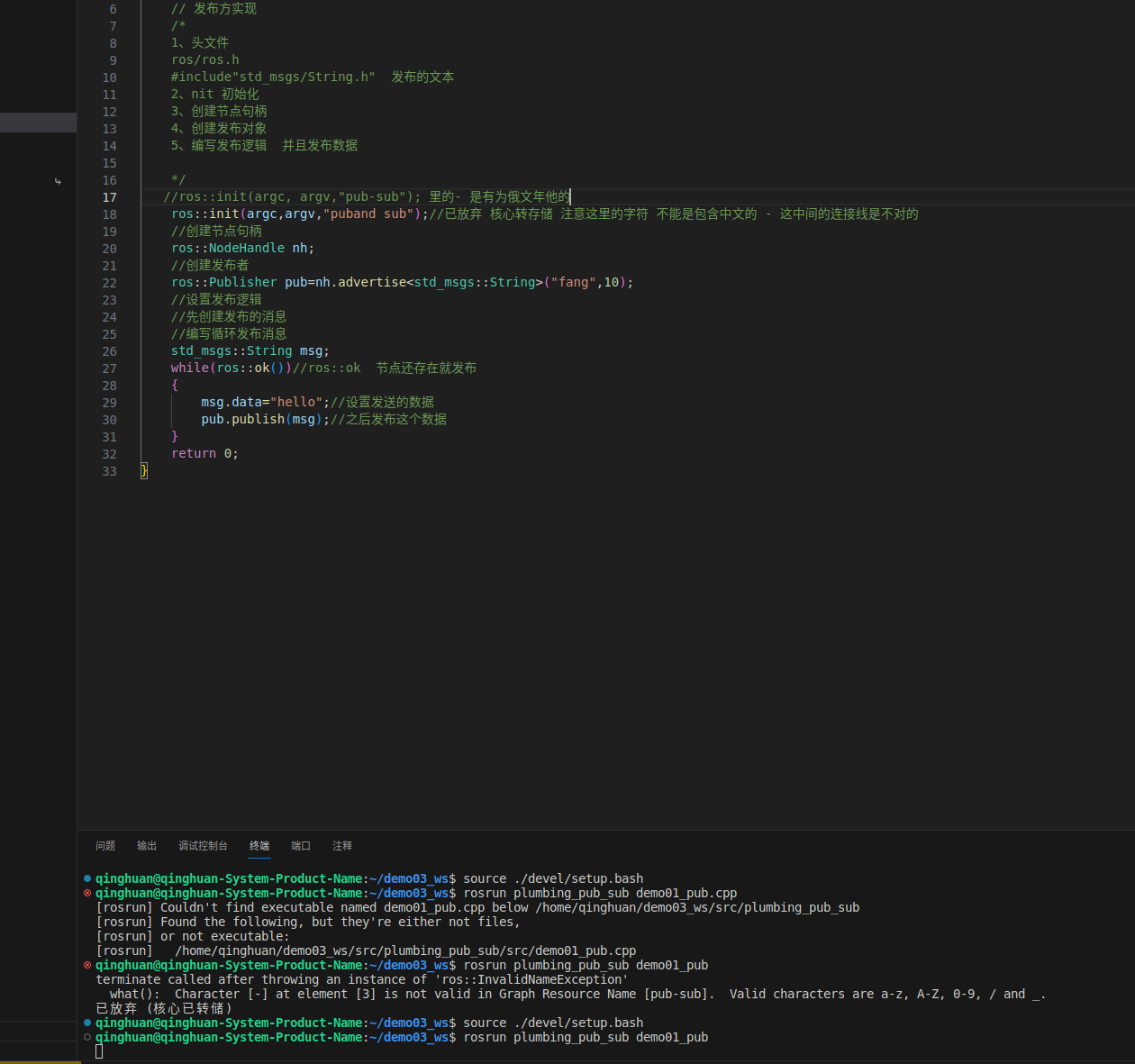
queue\_size相当于 缓冲区的大小



问题记录

ros::init(argc, argv,"pub-sub"); 里的- 是有中文的 字符 不可以

ros::init(argc,argv,"puband sub");//已放弃 核心转存储 注意这里的字符 不能是包含中文的 - 这中间的连接线是不对的

rostopic echo 话题名称

rostopic echo fang

在终端打印出发布的信息

编写发布逻辑

假设：以0.1s发布一次数据 每次发送要有数据

/要求以10hz发送数据

//ros::Rate rate(10)发送频率

//构造函数 给个数据 发送个数据

//调用rate的sleep函数 让这个睡 0.1s 10hz 0.1s一次

ros::Rate rate(10);

//设置编号

int count = 0;

while(ros::ok())

{

//字符串边界

std::stringstream ss;

//拼接这个字符

ss<<"hello --->"<<count;

//ss.str() 这个是stringstream str()函数可以把拼接的内容 字符拿出来

msg.data=ss.str();

pub.publish(msg);

//设置我发布的数据 python和c++都有这个%s的使用方法

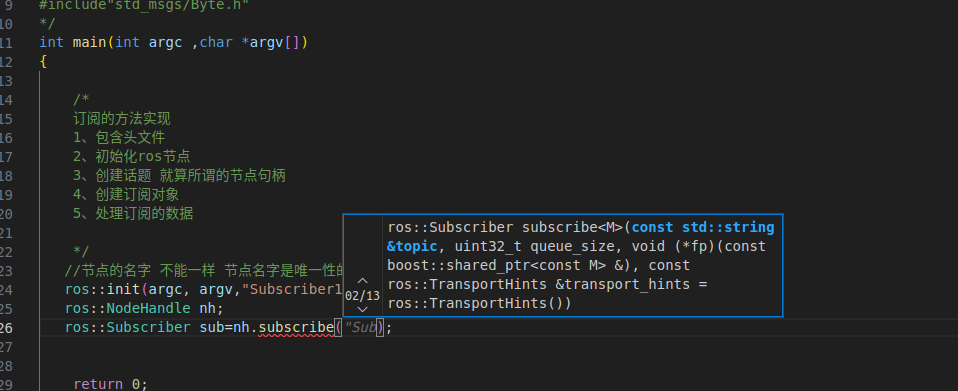
ROS\_INFO("我发布的数据 :%s",msg.data.c\_str());

//ctrl + z 暂停终端

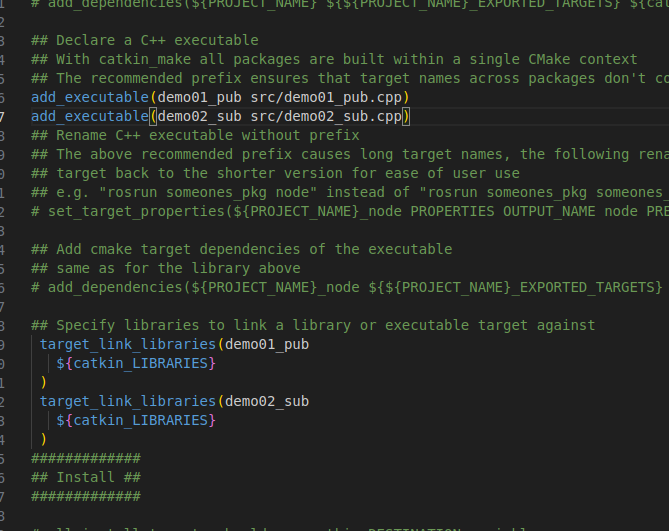
rate.sleep();

count++;

创建订阅者



多文件



source ./devel/setup.bash

rosrun plumbing\_pub\_sub demo01\_pub

source ./devel/setup.bash

rosrun plumbing\_pub\_sub demo02\_sub

