

rostopic 的操作

rostopic 是用于 ROS 节点间通信的工具

rostopic ping	测试到节点的连接状态
rostopic list	列出活动节点
rostopic info	打印节点信息
rostopic machine	列出指定设备上节点
rostopic kill	杀死某个节点
rostopic cleanup	清除不可连接的节点

Copy

- rostopic ping

测试到节点的连接状态

- rostopic list

列出活动节点

- rostopic info

打印节点信息

- rostopic machine

列出指定设备上的节点

- rostopic kill

杀死某个节点

- rostopic cleanup

清除无用节点，启动乌龟节点，然后 ctrl + c 关闭，该节点并没被彻底清除，可以使用 cleanup 清除节点

显示我们使用的方法

```
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rostopic
rostopic is a command-line tool for printing information about ROS Nodes.

Commands:
  rostopic ping      test connectivity to node
  rostopic list      list active nodes
  rostopic info      print information about node
  rostopic machine   list nodes running on a particular machine or list machines
  rostopic kill      kill a running node
  rostopic cleanup   purge registration information of unreachable nodes

Type rostopic <command> -h for more detailed usage, e.g. 'rostopic ping -h'
```

rostopic ping/list -h

rostopic 是那一组指令下的 某一个方法 -h 就算 help 显示帮助文档

这个是节点的名字

```
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rostopic list
/msgpub
/newmsg_sub
/rosout
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$
```

区分节点和话题的名字

节点名字是 `ros::init(argc,argv,节点的名字 string 类型具有唯一性`

话题的名字 是 `nodehandle 的 nh 对象下的 advertise 中 消息类型`

下面这两个图 就显示了 节点的名字和话题名字的区别

```
int main(int argc,char *argv[])
{
    setlocale(LC_ALL,"");

    ros::init(argc,argv,"newmsg_sub"); // 节点名字
    ROS_INFO("this is a subscriber");
    ros::NodeHandle nh;
    //话题名称 队列长度 回调函数
    ros::Subscriber sub=nh.subscribe("newmsg",10,donewMsg);
    ros::spin();//每次到这个地方 都去执行回调函数
    return 0; // 话题
}
```

```
int main(int argc,char *argv[])
{
    setlocale(LC_ALL,"");
    ros::init(argc,argv,"msgpub");
    ros::NodeHandle nh;
    // 那个作用域下的 文件 是这个include 是 plumbing_pub_sub作用域下的Person
    ros::Publisher pub=nh.advertise<plumbing_pub_sub::Person>("newmsg",10);
    //表明自己建立的msg文件 之后给他付初值
```

rostopic ping /节点名称 显示了连接的状态
节点的名字 我们延时了 3s 我们之间的依赖

```
/rosout
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rostopic ping /msgpub
rostopic: node is [/msgpub]
pinging /msgpub with a timeout of 3.0s
xmlrpc reply from http://qinghuan-System-Product-Name:37567/    time=9.846687ms
xmlrpc reply from http://qinghuan-System-Product-Name:37567/    time=0.413418ms
xmlrpc reply from http://qinghuan-System-Product-Name:37567/    time=0.401497ms
xmlrpc reply from http://qinghuan-System-Product-Name:37567/    time=0.404835ms
```

rostopic info /节点名字
显示了相关信息 以及我们用的 newmsg 的文件 话题的名字 newmsg 之类的
之后我们 newmsg 是我们的 node 节点的名字

```
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rostopic info /newmsg_sub
-----
Node [/newmsg_sub]
Publications:
* /rosout [rostopic_msgs/Log]

Subscriptions:
* /newmsg [plumbing_pub_sub/Person]

Services:
* /newmsg_sub/get_loggers
* /newmsg_sub/set_logger_level

contacting node http://qinghuan-System-Product-Name:41617/ ...
Pid: 35893
Connections:
* topic: /rosout
  * to: /rosout
  * direction: outbound (33907 - 127.0.0.1:45284) [13]
  * transport: TCPROS
* topic: /newmsg
  * to: /msgpub (http://qinghuan-System-Product-Name:37567/)
  * direction: inbound (47416 - qinghuan-System-Product-Name:40129) [12]
  * transport: TCPROS

qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$
```

rostopic echo 句柄名称（或者是话题） 可以实现 将发送到这个句柄（话题）的信息打印出来
rostopic list 看所有的话题 也就是 nodehandle 创建的巨笔

rosservice call addInts

```
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosservice call addInts "num1:10
num2:10"
ERROR: Incompatible arguments to call service:
Not enough arguments:
* Given: ['num1:10 num2:10']
* Expected: ['num1', 'num2']
Provided arguments are:
* num1:10 num2:10 (type str)

Service arguments are: [num1 num2]
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosservice call addInts "num1: 2
num2: 21"
sum: 23
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$
```

addInts 是服务的名称

```
//创建函数句柄
ros::NodeHandle nh;
//nh.serviceClient 创建客户
ros::ServiceClient client= nh.serviceClient<plumbing_server_client::AddInts>("addInts");
//声明数据
plumbing_server_client::AddInts ai;
// requests plumbing_server_client::AddInts::Request & req; 这个是我们的request调用的法那个法 我们需要调用这个参数
ai.request.num1 = atoi(argv[1]);
ai.request.num2 = atoi(argv[2]);

//向客户端发送数据之前 我们先 看server是否启动 没有启动则挂起
/*
client.waitForExistence();
ros::service::waitForService("addInts"); 传的参数是被等待的 服务 或者说 是具体的节点
这两个函数都可以实现 对 通信的挂起
*/
//client.waitForExistence();
```

```
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$ rosrn plumbing_server_client demo01_server
[ INFO] [1710671130.252628197]: 服务器端启动
[ INFO] [1710671320.429561341]: 收到的请求数据 num1 =2, num2=21
[ INFO] [1710671320.429607751]: 求和的结果是 sum = 23
^Z
[2]+ 已停止                  rosrn plumbing_server_client demo01_server
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$
```