rosnode 的操作

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

Copy

- rosnode ping

  测试到节点的连接状态

- rosnode list

  列出活动节点

- rosnode info

  打印节点信息

- rosnode machine

  列出指定设备上的节点

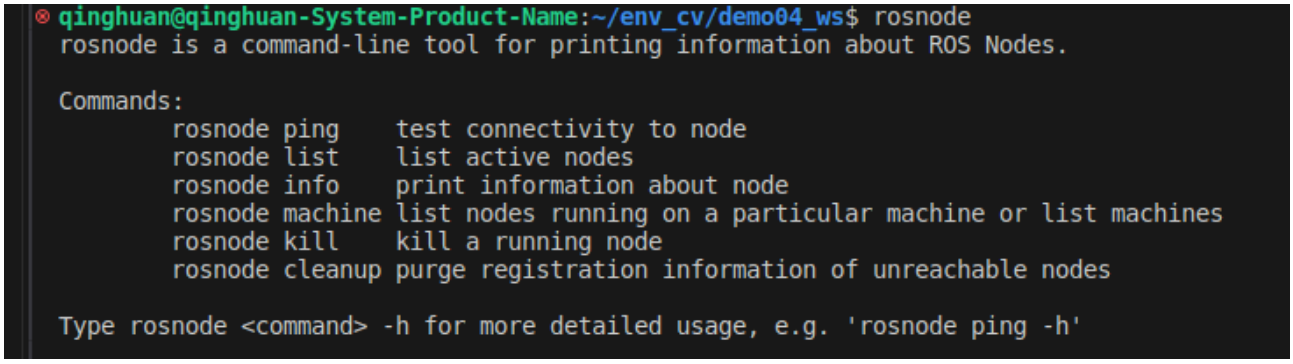
- rosnode kill

  杀死某个节点

- rosnode 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



rosnode ping/list -h
rosnode 是那一条据指令下 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$ rosnode 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");
    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;
}
```

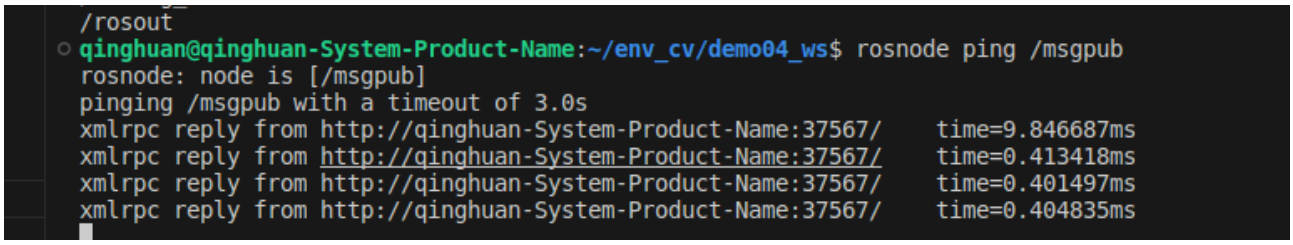
节点名字

话题

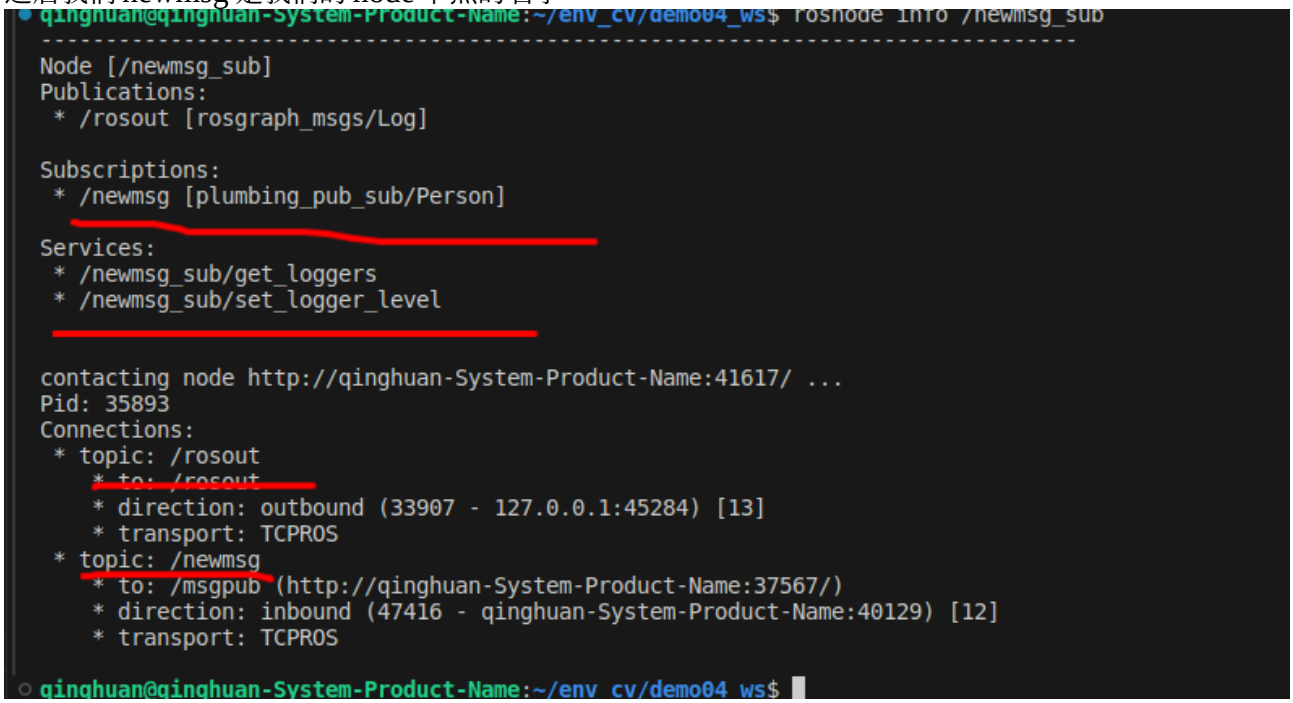
```
int main(int argc,char *argv[])
{
    setlocale(LC_ALL,"");
    ros::init(argc,argv,"msgpub");
    ros::NodeHandle nh;
    // 那个作用域下的 文件 是这个include 是 plumbing_pub_sub作用 羽 下的Person
    ros::Publisher pub=nh.advertise<plumbing_pub_sub::Person>("newmsg",10);

    //声明自己建立的msg文件 之后给他付初值
```

rosnode 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



rosnode info /节点名字
显示了相关信息 以及我们用的 nwsmsg 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



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$ rosrun 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               rosrun plumbing_server_client demo01_server
qinghuan@qinghuan-System-Product-Name:~/env_cv/demo04_ws$
```