Action讲义

### 背景知识

1.1 回顾action机制

什么是action，回顾第一个action的例子，这里主要是action的第一个dishwasher

在之前的character3（第三章）中，我们引入了ROS通讯机制，主要是topic, service和action，其中最主要的是topic（话题）和topic的格式msg，对于action部分，因为篇幅的关系，介绍的比较有限，当涉及到机械臂的通讯关系时，action服务会更加适用，这里先来回顾一下之前的内容：

在授课中提到什么是action以及.action格式，action是一种通讯方式，他的原理也是基于ros的msg基础上，正如之前的这张原理图

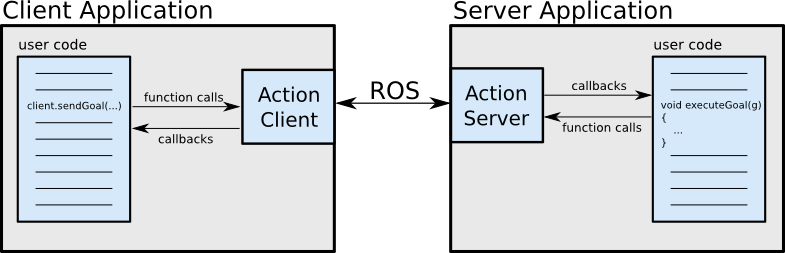


图1

这张图里我们可以看到服务和用户的两个板块用ROS连接起来，这就是ros中定义的message信息，Action是在使用多个msg消息封装以后的ros通讯方式，其本质仍是message，特点在于这种机制的构建是比较困难的，所以做成了封装后的action通讯模块。action的格式和功能有一张图会比较清楚的解释

客户端action

client

服务器action

service

goal

cancel

status

result

feedback

从用户端向服务器

从服务器向用户端

图

这里的五个箭头代表了5个message，分别是

1. Goal ：由客户端发送给服务端的消息，用来指定服务端需要完成的内容，比如指定机械臂下一时刻的全局位置；
2. Cancel：发送后，取消执行当前命令的消息
3. Status：返回服务器的当前状态
4. Result：当goal完成时，由服务器向客户器发送的消息，特点是只发送一次。比如说他返回的结果是当前的位置坐标信息。
5. Feedback：服务器告诉客户端goal完成程度的消息，表示的是当前的进度。

为什么是action

1.2 action的文件格式

在之前的topic和service中我们已经知道了它们对应的通讯格式是什么，同样的作为通讯机制，aciton也有自己的格式，其文件格式为\*.action，其中\*代表任意个字符，我们来回顾一下曾经的aciton文件写法。

代码来自于DoDishes.action文件

# Define the goal

Unit32 dishwasher\_id # Specify which dishwasher we want to use

---

# Define the result

Uint32 total\_dishes\_cleaned

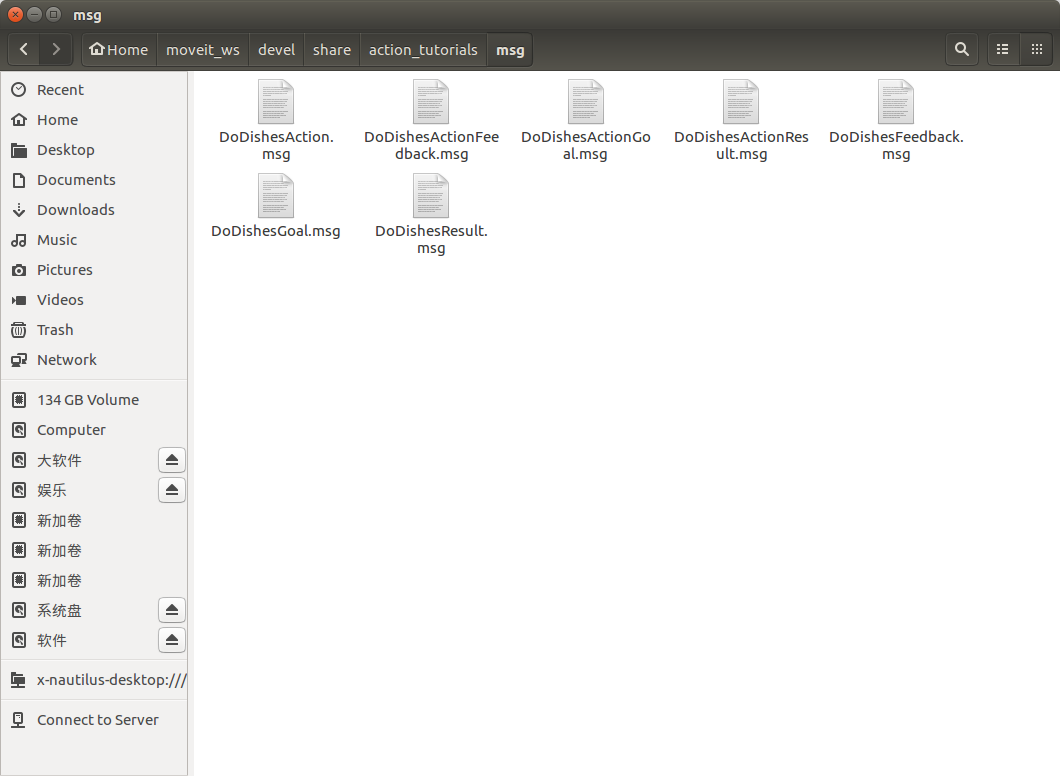
---

# Define the feedback\_message

Float32 percent\_complete

这个之前使用过的例子中主要是goal, result和feedback这三项，事实上这三项在编译后会生成7个msg文件

* DoDishesAction.msg
* DoDishesActionGoal.msg
* DoDishesActionResult.msg
* DoDishesActionFeedback.msg
* DoDishesGoal.msg
* DoDishesResult.msg
* DoDishesFeedback.msg



图

这7个msg文件的后三个没有action尾部的文件相当于将定义在.action 中的message分成了三个message，而带有action后缀的相当于编译成了对应的头文件格式，第一个文件是对于它们三对之间的对应关系描述，这里的通讯机制可以不必深究，只需要明白其内部相当于自定义了三个message就可以了。这一点很重要，因为后期的机械臂中有很多的自定义mes-

sage方式。

1.3 三种通讯机制的对比

为什么需要action这种方式，就需要区别topic，service和action的不同，借此说明aciton的特点。在action的格式文件中也已经看到，action文件中一般是三项goal, result和feedback，相对于topic方式，他多了result和feedback这两个需要由服务端返回客户端的消息；相对于service方式，他又多了feedback这个实时的消息，从这里就可以看出他的特点在于实时的回馈运行状态。

下面通过一个表格来区别三种方式：

|  |  |  |  |
| --- | --- | --- | --- |
| 机制 | Topic | Service | Action |
| 格式 | .msg | .srv | .action |
| 方式 | 单向发送 | 有请求，然后反馈 | 有目标，有结果并且有进度反馈 |
| 通讯协议 | TCP/IP | TCP/IP | TCP/IP |
| 实时性 | 异步 | 同步 | 同步 |

可以看出，action方式就相当于增强过的topic和service，它最大的好处就是能够得到实时的运行反馈值，对于机械臂而言就可以得到末端执行件或者各关节的运行状况，这对于机械臂而言是极其重要的，同时这里也可以看到action方式，其本质是多个message消息的封装，所以我们也可以采取自定义message的方式，来创建适应自己需求的消息格式。

1.4 对DoDishes的回顾

（内容待完善，这里主要是重复action client和service的建立过程）

### action练习1

第一个action的应用例子，这是一个重复性联系，文件来源于github的一个Python文件

https://github.com/NickKnack15/python-actionlib-demo

### Action练习2

第二个action的应用例子，最好能牵扯到一点机械臂的内容

这里需要联系moveit的内容阐释action的用法，同时会引入自定义message在moveit中的用法，内容同样待补充

<http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/setup_assistant/setup_assistant_tutorial.html>

参考moveit的官网内容，完成这一部分