链滴

[ROS]创建srv和msg

作者：lixiang0

原文链接：https://ld246.com/article/1472116323878

来源网站：链滴

前言：本教程介绍了如何创建和构建msg和srv文件，并介绍了rosmsg，rossrv和roscp命令行工具。

####1.Introduction to msg and srv

msg：msg文件是简单的文本文件，它描述了ROS消息的属性。这些文件被用来生成消息在不同语言的源码。

srv：srv文件描述了服务。它由2部分组成：请求和响应。

msg文件保存在包的msg字典中。srv文件保存在srv字典中。

msg文件是简单的文本文件，它的每一行包含一个属性类型和属性名字。下面是属性的类型：

```
int8, int16, int32, int64 (plus uint*)
float32, float64
string
time, duration
other msg files
variable-length array[] and fixed-length array[C]
```

ROS中有一种特殊的类型：Header。Header包含一个邮戳和ROS中普遍使用的坐标帧信息。你将频的看到msg文件的第一行是Header header。

这里给出一个使用msg的例子，它用到了Header，string，两个其他的msg：

```
  Header header
  string child_frame_id
  geometry_msgs/PoseWithCovariance pose
  geometry_msgs/TwistWithCovariance twist
```

srv和msg很类似，区别在于他们包含的两部分：请求和响应。两个部分之间通过单独的‘---’一行分。这里给出一个srv的例子：

```
int64 A
int64 B
---
int64 Sum
```

上面的例子中A和B是请求，Sum是响应。

####2.Using msg

#####2.1Creating a msg

让我们在之前创建的包中定义一个新的msg。

```
$ cd ~/catkin_ws/src/beginner_tutorials
$ mkdir msg
$ echo "int64 num" > msg/Num.msg
```

示例中的.msg文件只包含了一行。当然也可以通过一行添加一个复杂的元素来创建更复杂的文件，就下面的那样：

```
string first_name
string last_name
uint8 age
```

```
uint32 score
```

接下来，我们需要确定这个msg文件要转换为那种源码，是C++，Python，还是别的什么语言：

打开package.xml文件，确保下面的几行是在文件中，并且取消注释：

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

文件中注明在构建的时候，需要"message_generation"，运行的时候需要"message_runtime".

打开CMakeList.txt文件，在文件中添加message_generation依赖到已有的find_package标签中。可以简单的添加message_generation到COMPONENTS列表中，添加之后为：

```
# Do not just add this to your CMakeLists.txt, modify the existing text to add message_genera
ion before the closing parenthesis
find_package(catkin REQUIRED COMPONENTS
   roscpp
   rospy
   std_msgs
   message_generation
)
```

你可能已经注意到很多时候就算你不调用find_package添加所有依赖，你的工程也可以构建成功。这因为catkin联合了你的所有工程到一个工程里面，所以如果之前的工程里面调用了find_package，之的工程也使用同样的配置。但是忘记调用find_package在单独的构建工程的时候非常容易失败。

同样的也需要确定导出运行时依赖的消息。

```
catkin_package(
  ...
  CATKIN_DEPENDS message_runtime ...
  ...)
```

找到下面部分的代码：

```
# add_message_files(
#   FILES
#   Message1.msg
#   Message2.msg
# )
```

通过删除#符号取消注释，然后替换Message×.msg文件为你的.msg文件，完成之后看起来像：

```
add_message_files(
  FILES
  Num.msg
)
```

通过手动的添加.msg文件，我们可以确定CMake知道在你添加了其他的.msg文件之后需要重新配置程。

现在我们需要确定generate_messages()方法被调用了。

对于ROS Hydro及其之后的版本, 取消注释下面的几行:

```
# generate_messages(
#   DEPENDENCIES
#   std_msgs
# )
```

修改之后为:

```
generate_messages(
  DEPENDENCIES
  std_msgs
)
```

在更早的版本，只需要取消注释下面这一行:

```
generate_messages()
```

现在我们已经准备好了根据msg的定义生成源文件.如果你想立马开始做，忽略接下来的部分，直接跳Common step for msg and srv。

####3.Using rosmsg

这就是创建一个msg所有需要做的。让我们来确保这个msg对于ROS是可见的，使用rosmsg show可。

用法:

```
$ rosmsg show [message type]
```

例如:

```
$ rosmsg show beginner_tutorials/Num
```

将看到:

```
int64 num
```

在之前的例子中，消息的类型由2部分组成:

beginner_tutorials -- 消息定义在哪个包

Num -- Num是消息的名称.

如果你不记得是msg是在具体哪个包里面，你可以不指定包名。尝试:

```
$ rosmsg show Num
```

可以看到:

```
[beginner_tutorials/Num]:
int64 num
```

####4.Using srv

#####4.1Creating a srv

首先进入我们之前创建的包去创建一个srv:

```
$ roscd beginner_tutorials
```

```
$ mkdir srv
```

这次我们不手动的创建一个新的srv定义，而是从别的包中拷贝一个现有的。

为了达成我们的目的，roscp是一个非常有用的命令行工具，它可以用来从一个包中拷贝文件到另一包。

方法:

```
$ roscp [package_name] [file_to_copy_path] [copy_path]
```

现在我们从rospy_tutorials包中拷贝一个服务：

Now we can copy a service from the rospy_tutorials package:

```
$ roscp rospy_tutorials AddTwoInts.srv srv/AddTwoInts.srv
```

接下来，我们需要确定这个srv文件要转换为那种源码，是C++，Python，还是别的什么语言：

除非你已经做好了，不然打开package.xml文件，然后确保下面两行没有被注释掉：

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

像之前那样，在构建的时候需要message_generation,而在运行的时候，需要message_runtime。

除非在之前已经在创建新的msg的时候做好了，不然接下来需要在CMakeList.txt文件中添加message generation依赖：

```
# Do not just add this line to your CMakeLists.txt, modify the existing line
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
 message_generation
)
```

（message_generation对msg和srv都生效，所以这里可以不用管名字重复的问题）

同样的你需要对package.xml做之前在创建msg的时候同样的事情，所以我们向上看看额外需要的依：

删除#取消对下面几行的注释:

```
# add_service_files(
#   FILES
#   Service1.srv
#   Service2.srv
# )
```

然后把Service*.srv这几个文件改成你的文件:

```
add_service_files(
  FILES
  AddTwoInts.srv
)
```

现在我们已经准备好了根据服务的定义生成源码了。如果你想立马做的话，跳到Common step for m g and srv。

#####4.2.Using rossrv

这就是创建一个srv需要做的全部事情。让我们通过rossrv来确定ROS可见我们刚才创建的srv。
用法:

```
$ rossrv show <service type>
```

例如:

```
$ rossrv show beginner_tutorials/AddTwoInts
```

你将看到:

```
int64 a
int64 b
---
int64 sum
```

跟rosmsg类似, 可以找到需要的srv文件，而不需要指定包名:

```
$ rossrv show AddTwoInts
[beginner_tutorials/AddTwoInts]:
int64 a
int64 b
---
int64 sum

[rospy_tutorials/AddTwoInts]:
int64 a
int64 b
---
int64 sum
```

####5.Common step for msg and srv

确定CMakeLists.txt文件中下面的几行没有被注释 :

```
# generate_messages(
#   DEPENDENCIES
# #  std_msgs  # Or other packages containing msgs
# )
```

取消注释，然后添加使用msg需要依赖的任何.msg文件，完成之后看起来像下面的样子:

```
generate_messages(
  DEPENDENCIES
  std_msgs
)
```

现在已经在包中添加了一些新的msg，重新make包:

```
# In your catkin workspace
```

```
$ cd ../..
$ catkin_make install
$ cd -
```

任何在msg文件夹中的.msg文件将生成所有能被支持的语言的源码。C++消息头文件生成在~/catkin ws/devel/include/beginner_tutorials。Python脚本创建在~/catkin_ws/devel/lib/python2.7/dist packages/beginner_tutorials/msg。lisp文件生成在~/catkin_ws/devel/share/common-lisp/ros/ eginner_tutorials/msg/。

类似的，任何srv目录中的.srv文件也能生成被支持的语言的源码。对于C++，生成在msg的头文件相的目录中。对于Python和Lisp，生成在msg目录里的srv目录。

所有消息格式的描述参考：http://wiki.ros.org/ROS/Message_Description_Language.

如果你使用新的msg构建c++节点，你需要在node和message之间声明依赖，参考：http://docs.ros org/hydro/api/catkin/html/howto/format2/building_msgs.html。

####6.Getting Help

如今已经有了很多的新ROS工具。很难去记得每一个命令需要什么参数，幸运的是，大多数的ROS工提供了他们自己的帮助。

运行：

```
$ rosmsg -h
```

你可以看到不同的rosmsg子命令.

```
Commands:
  rosmsg show Show message description
  rosmsg users  Find files that use message
  rosmsg md5  Display message md5sum
  rosmsg package  List messages in a package
  rosmsg packages List packages that contain messages
```

也可以查看子命令的帮助：

```
$ rosmsg show -h
```

这个命令显示rosmsg show需要那种参数:

```
Usage: rosmsg show [options] <message type>

Options:
  -h, --help  show this help message and exit
  -r, --raw   show raw message text, including comment
```

####7.Review

让我们列出一下我们目前位置使用到的命令:

```
rospack = ros+pack(age) : 提供包的信息
roscd = ros+cd : 进入目标包的目录
rosls = ros+ls : 列出包的文件
roscp = ros+cp : 从一个包拷贝文件到另外一个包
rosmsg = ros+msg : message的定义信息
rossrv = ros+srv : service的定义信息
```

catkin_make : 编译一个包
rosmake = ros+make : 编译一个包（不在catkin工作目录的时候）

####8.Next Tutorial

接下来我们将学习编写一个简单的发布器和订阅器。

原文链接：[ROS]创建srv和msg