理解 ROS rqt_console 和 roslaunch

AI 航 团队

这篇教程将介绍使用 rqt_console 和 rqt_logger_level 来调试以及使用 roslaunch 一次启动许多 nodes.如果你使用 ROS fuerte 或者更早的版本，rqt 不是十分完善，请查看这篇文章使用基于 old rxthis page.

1.前提 rqt 和 tuetlesim package

需要用到 rqt 和 tuetlesim package．如果没有安装，请执行：

```
$ sudo apt-get install ros-<distro>-rqt
```

ros-<distro>-rqt-common-plugins ros-<distro>-turtlesim

注意：前面的教程中已经编译过 rqt 和 tuetlesim 这两个 package 了，如果不确定，再安装一次也无妨.

2. 使用 rqt_console 和 rqt_logger_level

rqt_console 附着在 ROS logging 框架上去显示 nodes 的输出结果.

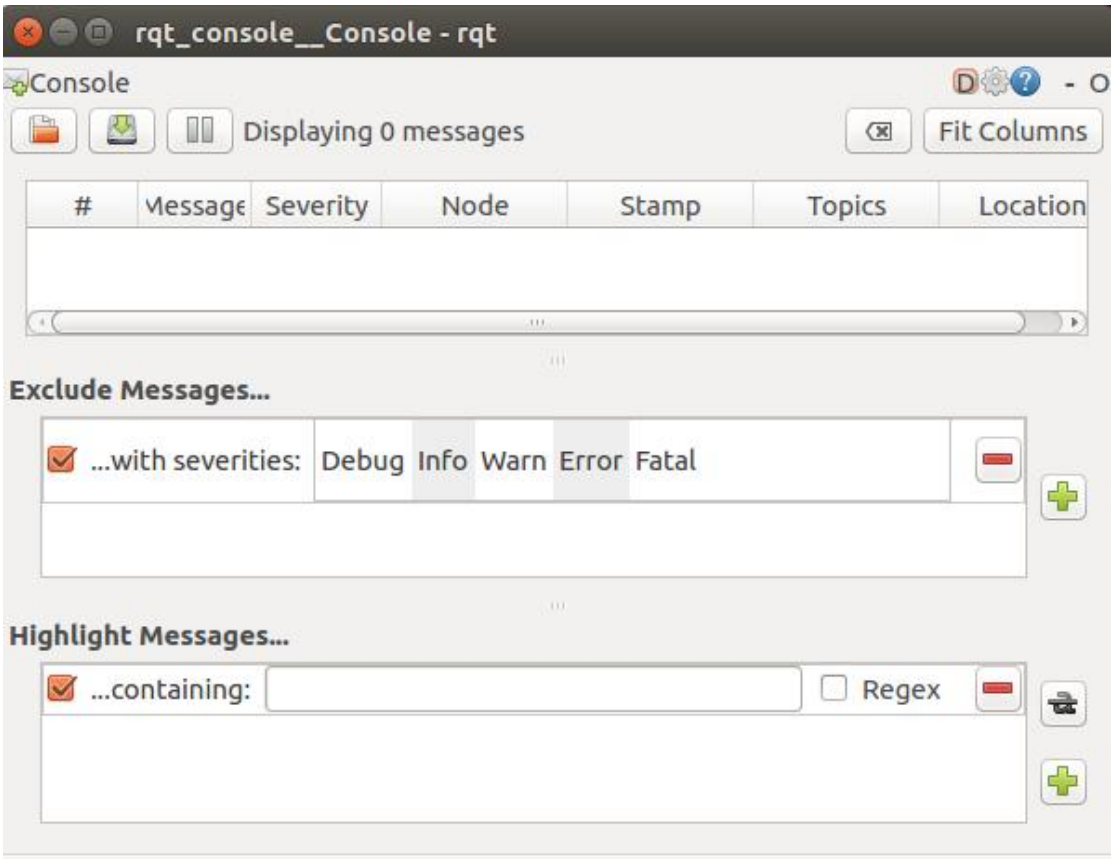
rqt_logger_level 允许我们去改变 nodes 运行时候的信息显示级别（调试，警告，信息和错误）。现在让我们看看 turtlesim 在 rqt_console 上的输出并且当我们使用 turtlesim 的时候变化 logger 级别。

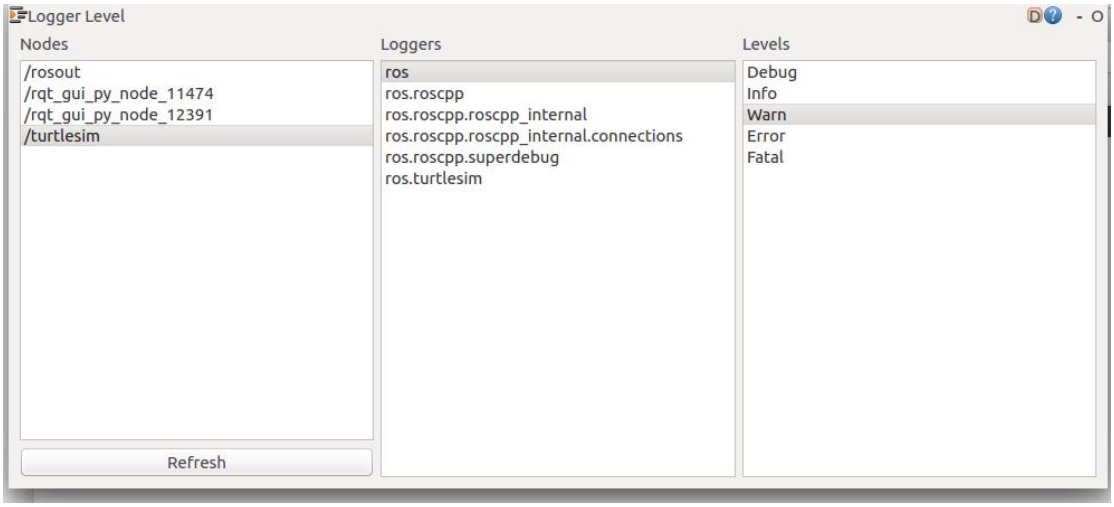
在运行 turtlesim 之前，在两个新的终端中分别运 rqt_consolerqt 和 rqt_logger_level:

```
$ rosrun rqt_console rqt_console
```

```
$ rosrun rqt_logger_level rqt_logger_level
```

你会看到两个弹出的窗口：

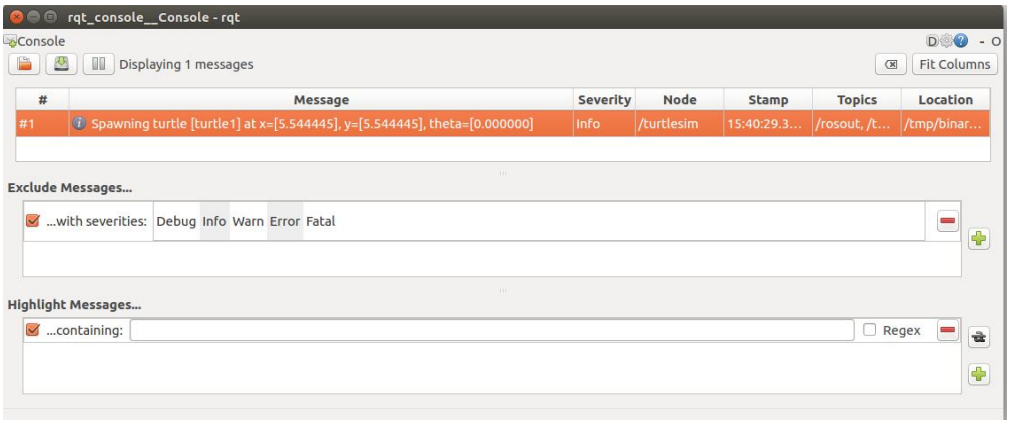




现在再在新的窗口中运行：

```
$ rosrun turtlesim turtlesim_node
```

因为默认的记录器级别是 INFO 所以你会看到 turtlesim 启动时发布的信息,大概是这个样子：

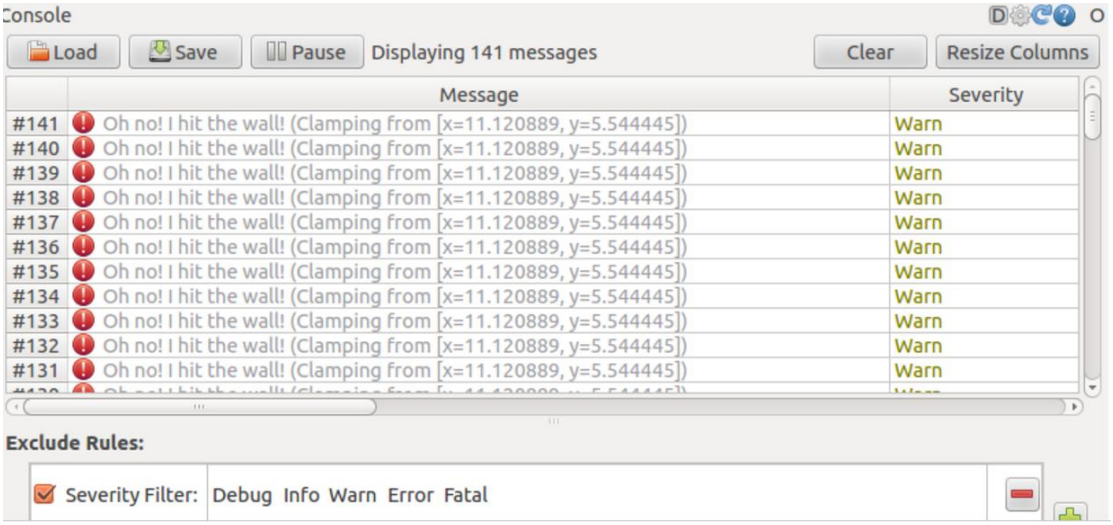


现在我们把记录器级别改为 Warn，在 rqt_logger_level 窗口中刷新 nodes 并且选择 Warn 作为显示选项:

现在把小乌龟遥控到墙边看看在 rqt_console 上有什么显示：

对于 ROS Hydro 和之后的版本：

```
rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 0.0]'
```



2.1 logger 级别的概述

记录级别是按下列的的优先级别区分的：

Fatal

Error

Warn

Info

Debug

Fata 的级别最高，Debug 的级别最低．通过设置 logger 级别，你会得到　这个优先级别或更高级别的 message.比如，通过设置级别为 Warn,我们会得到所有的 Warn,Error,和 Fatal 的记录消息.

先 ctrl+c turtlesim，并且用 roslaunch 去生成更多的 turtlesim nodes 和一个 mimicking node，让一个 turtlesim 去模仿另一个。

2.2 使用 roslaunch

roslaunch 按照 launch 文件中的定义去启动 nodes．

用法：

```
roslaunch [package] [filename.launch]
```

首先进入我们之前创建和编译的 beginner_tutorials package：

```
$ roscd beginner_tutorials
```

如果 roscd 说类似于：No such package/stack 'beginner_tutorials' 你需要启动环境变量设置的文件，像你之前在 create_a_workspace 教程末尾中做的一样。

2.3 launch 文件

现在创建一个叫做 turtlemimic.launch 的 launch 文件并且把下面的东西粘贴在上面：

```
<launch>

 <group ns="turtlesim1">

   <node pkg="turtlesim" name="sim" type="turtlesim_node"/>

 </group>

<group ns="turtlesim2">

 <node pkg="turtlesim" name="sim" type="turtlesim_node"/>

</group>

 <node pkg="turtlesim" name="mimic" type="mimic">
```

2.4 Launch 文件的解释

现在我们把 xml 分解：

```
1 <launch>
```

我们用 lauch 标签开始 launch 文件，所以 launch 文件是这样鉴定的.

```
 <group ns="turtlesim1">
   <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
 </group>
<group ns="turtlesim2">
 <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
</group>
```

我们用一个叫做 sim 的 turtlesim node 定义两个命名空间 turtlesim1 和 turtlesim2，

这样我们就可以启动两个仿真器而不会有名字冲突了。

```
<node pkg="turtlesim" name="mimic" type="mimic">
  <remap from="input" to="turtlesim1/turtle1"/>
  <remap from="output" to="turtlesim2/turtle1"/>
</node>
```

我们通过把 topic 的输入和输出去重命名为 turtlesim1 和 turtlesim 2 来定义 mimic node(即 messages 在 topic 中从 turtlesim1 输入，从 turtlesim 2 输出），这样重命名会导致 turtlesim 2 模仿 turtlesim1。

```
16 <launch>
```

末尾的 xml 标签也是代表 launch 文件。

2.5 roslaunching

现在我们用 roslaunch 启动 launch 文件：

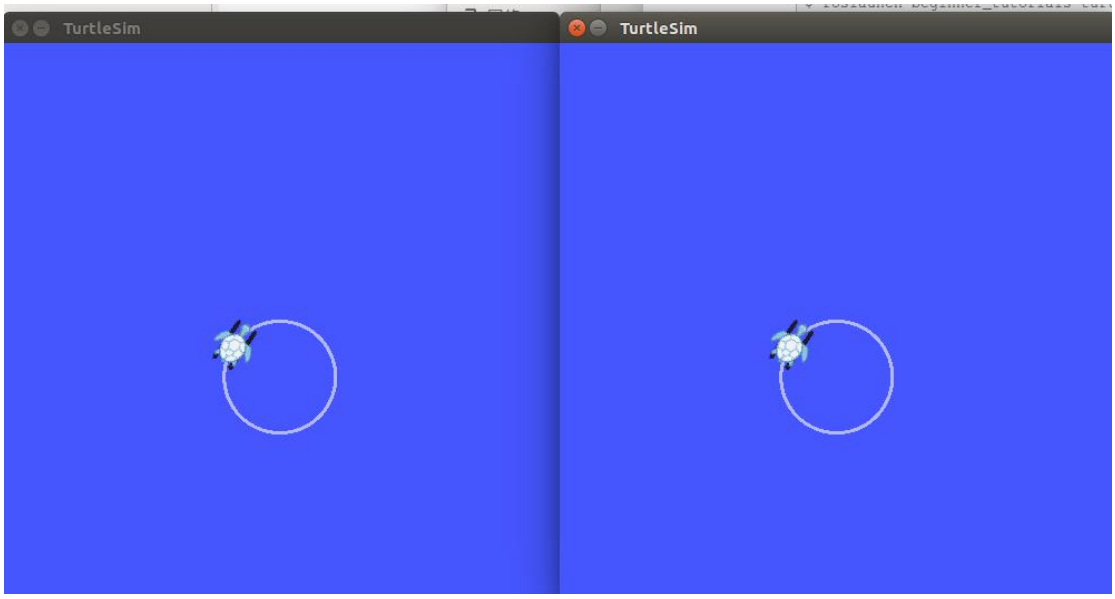
```
roslaunch beginner_tutorials turtlemimic.launch
```

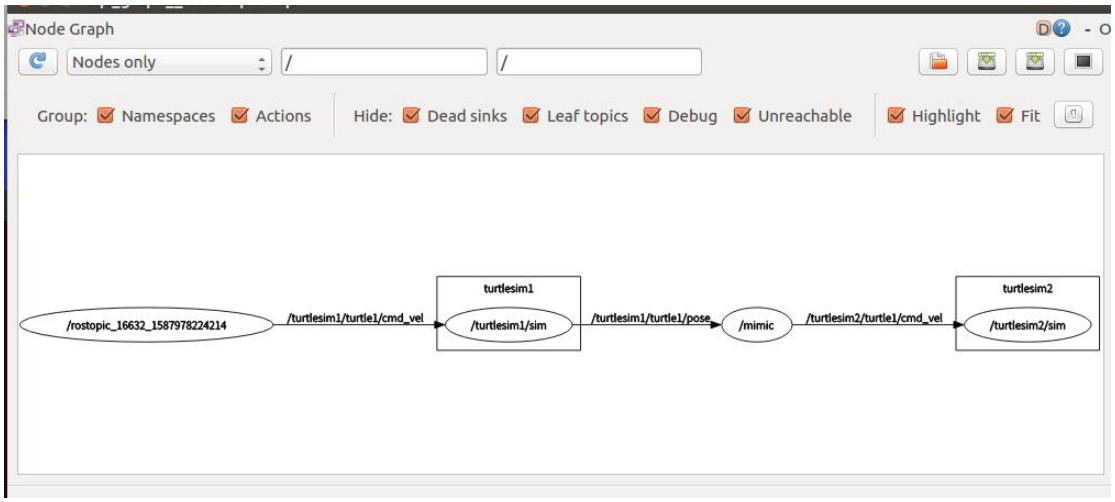
两个 turtlesim 会启动，在新的终端启动中并且发送 rostopic 命令：对于 ROS Hyhydro:

```
$ rostopic pub /turtlesim1/turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, -1.8]'
```

你将会看到即使命令只是发布给 turtlesim1 但是两个小乌龟都开始运动。

可以用 rqt_graph 去更好的理解 launch 文件做了什么．运行 rqt 的主窗口选择 rqt_graph：


TurtleSim
TurtleSim


Node Graph
Nodes only
/
/
Group: Namespaces Actions
Hide: Dead sinks Leaf topics Debug Unreachable
Highlight Fit
/rostopic_16632_1587978224214
/turtlesim1/turtle1/cmd_vel
turtlesim1
/turtlesim1/sim
/turtlesim1/turtle1/pose
/mimic
/turtlesim2/turtle1/cmd_vel
turtlesim2
/turtlesim2/sim