学习机器人操作系统（ROS）最好的参考资料就是 ROS Wiki：http://wiki.ros.org/cn，其中包括了 ROS 应该如何安装以及详细的中文入门教程，在使用 ROS 过程中常用的功能包说明以及遇到的常见问题也往往能在其中找到答案。本节对 ROS Wiki 入门教程进行了整理，帮助大家更快速的了解 ROS 相关核心概念。我们将从控制仿真小乌龟（turtlesim）出发，熟悉 ROS 中节点、话题、消息等概念，然后通过 Python 语言实现简单的订阅器和发布器编写，同时还会了解 ROS 中 rqt、rviz 等工具箱的使用。

理解 ROS 节点

- Nodes:节点,一个节点即为一个可执行文件，它可以通过 ROS 与其它节点进行通信。
- Messages:消息，消息是一种 ROS 数据类型，用于订阅或发布到一个话题。
- Topics:话题，节点可以发布消息到话题，也可以订阅话题以接收消息。
- Master:节点管理器，ROS 名称服务 (比如帮助节点找到彼此)。
- rosout: ROS 中相当于 stdout/stderr。
- roscore: 主机 + rosout + 参数服务器。

节点

一个节点其实只不过是 ROS 程序包中的一个可执行文件。ROS 节点可以使用 ROS 客户库与其他节点通信。节点可以发布或接收一个话题。节点也可以提供或使用某种服务。

（节点是 ros 中非常重要的一个概念，为了帮助初学者理解这个概念，这里举一个通俗的例子：

例如，咱们有一个机器人，和一个遥控器，那么这个机器人和遥控器开始工作后，就是两个节点。遥控器起到了下达指 令的作用；机器人负责监听遥控器下达的指令，完成相应动作。从这里我们可以看出，节点是一个能执行特定工作任务的工作单元，并且能够相互通信，从而实现一个机器人系统整体的功能。在这里我们把遥控器和机器人简单定义为两个节点，实际上在机器人中根据控制器、传感器、执行机构等不同组成模块，还可以将其进一步细分为更多的节点，这个是根据用户编写的程序来定义的。）

roscore

roscore 是你在运行所有 ROS 程序前首先要运行的命令。

请运行:

```
$ roscore
```

使用 rosnode

打开一个**新的终端**，可以使用 **rosnode** 像运行 roscore 一样看看在运行什么...

注意: 当打开一个新的终端时，你的运行环境会复位，同时你的~/.bashrc 文件会复原。如果你在运行类似于 rosnode 的指令时出现一些问题，也许你需要添加一些环境设置文件到你的~/.bashrc 或者手动重新配置他们。

rosnode 显示当前运行的 ROS 节点信息。rosnode list 指令列出活跃的节点:

```
$ rosnode list
```

- 你会看到:
- /rosout

这表示当前只有一个节点在运行: rosout。因为这个节点用于收集和记录节点调试输出信息，所以它总是在运行的。

rosnode info 命令返回的是关于一个特定节点的信息。

```
$ rosnode info /rosout
```

这给了我们更多的一些有关于 rosout 的信息.

现在，让我们看看更多的节点。为此，我们将使用 rosrun 弹出另一个节点。

使用 rosrun

rosrun 允许你使用包名直接运行一个包内的节点(而不需要知道这个包的路径)。

用法:

```
$ rosrun [package_name] [node_name]
```

现在我们可以运行 turtlesim 包中的 turtlesim_node。

在一个 **新的终端**:

```
$ rosrun turtlesim turtlesim_node
```

你会看到 turtlesim 窗口:

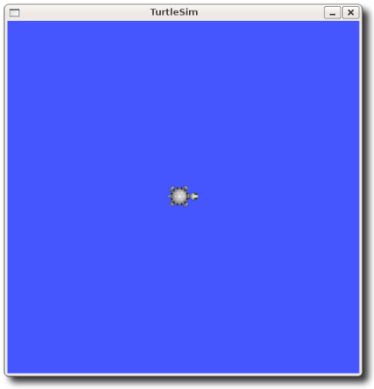


图 1：turtlesim 窗口

注意: 这里的 turtle 可能和你的 turtlesim 窗口不同。别担心，这里有许多版本的 turtle，而你的是一个惊喜!（一个可爱的小彩蛋 ~ ）

在一个 **新的终端**:

```
$ rosnode list
```

你会看见类似于:

```
/rosout
/turtlesim
```

ROS 的一个强大特性就是你可以通过命令行重新配置名称。

关闭 turtlesim 窗口停止运行节点 (或者回到 rosrun turtlesim 终端并使用`ctrl-c`)。现在让我们重新运行它，但是这一次改变节点名称:

```
$ rosrun turtlesim turtlesim_node __name:=my_turtle
```

现在，我们退回使用 rosnode list:

```
$ rosnode list
```

```
/rosout
/my_turtle
```

注意: 如果你仍看到 /turtlesim 在列表中，这可能意味着你在终端中使用 ctrl-c 停止节点而不是关闭窗口，或者你没有$ROS_HOSTNAME 环境变量。你可以尝试清除 rosnode 列表，通过: $ rosnode cleanup。

我们可以看到新的/my_turtle 节点。

理解 ROS 话题

roscore

首先确保 roscore 已经运行，打开一个新的终端:

```
$ roscore
```

如果你没有退出在上一篇教程中运行的 roscore，那么你可能会看到下面的错误信息:

- roscore cannot run as another roscore/master is already running.
- Please kill other roscore/master processes before relaunching

这是正常的，因为只需要有一个 roscore 在运行就够了。

turtlesim

在本教程中我们也会使用到 turtlesim，请**在一个新的终端中**运行:

```
$ rosrun turtlesim turtlesim_node
```

通过键盘控制 turtle

我们也需要通过键盘来控制 turtle 的运动，**请在一个新的终端中**运行:

```
$ rosrun turtlesim turtle_teleop_key
```

- [INFO] 1254264546.878445000: Started node [/teleop_turtle], pid [5528], bound on [aqy], xmlrpc port [43918], tcpros port [55936], logging to [~/ros/ros/log/teleop_turtle_5528.log], using [real] time
- Reading from keyboard
- ---------------------------
- Use arrow keys to move the turtle.

现在你可以使用键盘上的方向键来控制 turtle 运动了。如果不能控制，请**选中 turtle_teleop_key 所在的终端窗口**以确保你的按键输入能够被捕获。

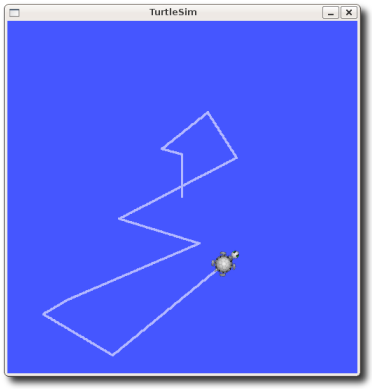


图 2：键盘控制 turtle

现在你可以控制 turtle 运动了，下面我们一起来看看这背后发生的事。

注意：如果遇到节点间无法通信，请在 home 中打开终端输入：gedit ~/.bashrc，注释掉：

export ROS_IP=`hostname -I | cut -d " " -f 1`。

ROS Topics

turtlesim_node 节点和 turtle_teleop_key 节点之间是通过一个 ROS **话题**来互相通信的。turtle_teleop_key 在一个话题上**发布**按键输入消息，而 turtlesim 则**订阅**该话题以接收该消息。下面让我们使用 rqt_graph 来显示当前运行的节点和话题。

使用 rqt_graph

在一个新终端中运行:

```
$ rosrun rqt_graph rqt_graph
```

或者:

```
$ rqt_graph
```

你会看到类似下图所示的图形:

图 3：ROS 节点话题关系图

如果你将鼠标放在/turtle1/cmd_vel 上方，相应的 ROS 节点（蓝色和绿色）和话题（红色）就会高亮显示。正如你所看到的，turtlesim_node 和 turtle_teleop_key 节点正通过一个名为 /turtle1/cmd_vel 的话题来互相通信。

图 4：节点和话题高亮显示

rostopic 介绍

rostopic 命令工具能让你获取有关 ROS **话题**的信息。

你可以使用帮助选项查看 rostopic 的子命令:

```
$ rostopic -h
```

- rostopic bw display bandwidth used by topic
- rostopic echo print messages to screen
- rostopic hz display publishing rate of topic
- rostopic list print information about active topics
- rostopic pub publish data to topic
- rostopic type print topic type

接下来我们将使用其中的一些子命令来查看 turtlesim。

使用 rostopic echo

rostopic echo 可以显示在某个话题上发布的数据。

用法:

```
rostopic echo [topic]
```

让我们在**一个新终端中**看一下 turtle_teleop_key 节点/turtle1/cmd_vel 话题上发布的数据。

```
$ rostopic echo /turtle1/cmd_vel
```

你可能看不到任何东西因为现在还没有数据发布到该话题上。接下来我们通过按下方向键使 turtle_teleop_key 节点发布数据。**记住如果 turtle 没有动起来的话就需要你重新选中 turtle_teleop_key 节点运行时所在的终端窗口。**

现在当你按下向上方向键时应该会看到下面的信息:

```
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

现在让我们再看一下 rqt_graph（你可能需要刷新一下 ROS graph）。正如你所看到的，rostopic echo(红色显示部分）现在也**订阅**了 turtle1/cmd_vel 话题。

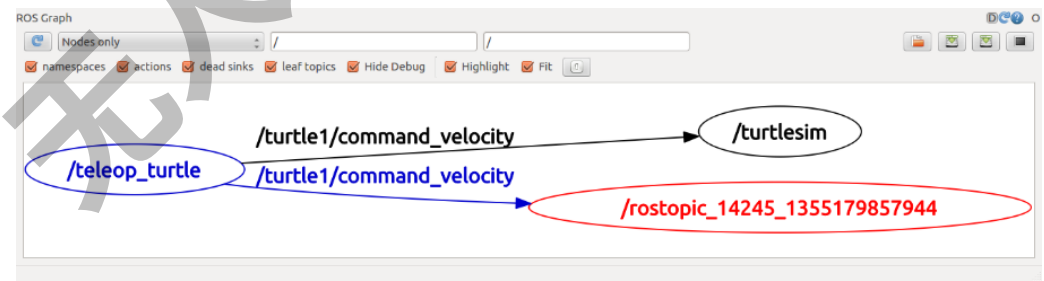


图 5：ROS 节点话题关系图

使用 rostopic list

rostopic list 能够列出所有当前订阅和发布的话题。

让我们查看一下 list 子命令需要的参数，在**一个新终端中**运行：

```
$ rostopic list -h
```

- Usage: rostopic list [/topic]
-
- Options:
- -h, --help show this help message and exit
- -b BAGFILE, --bag=BAGFILE
- list topics in .bag file
- -v, --verbose list full details about each topic
- -p list only publishers
- -s list only subscribers

在 rostopic list 中使用 **verbose** 选项:

```
$ rostopic list -v
```

这会显示出有关所发布和订阅的话题及其类型的详细信息。

ROS Messages

话题之间的通信是通过在节点之间发送 ROS **消息**实现的。对于发布器(turtle_teleop_key)和订阅器(turtulesim_node)之间的通信，发布器和订阅器之间必须发送和接收相同**类型**的消息。这意味着话题的**类型**是由发布在它上面的消息**类型**决定的。使用 rostopic type 命令可以查看发布在某个话题上的消息**类型**。

使用 rostopic type

rostopic type 命令用来查看所发布话题的消息类型。

用法:

```
rostopic type [topic]

$ rostopic type /turtle1/cmd_vel
```

你应该会看到:

```
geometry_msgs/Twist
```

我们可以使用 rosmsg 命令来查看消息的详细情况:

```
rosmsg show geometry_msgs/Twist
```

- geometry_msgs/Vector3 linear
- float64 x
- float64 y
- float64 z
- geometry_msgs/Vector3 angular
- float64 x
- float64 y
- float64 z

现在我们已经知道了 turtlesim 节点所期望的消息类型，接下来我们就可以给 turtle 发布命令了。

继续学习 rostopic

现在我们已经了解了什么是 ROS 的**消息**，接下来我们开始结合消息来使用 rostopic。

使用 rostopic pub

rostopic pub 可以把数据发布到当前某个正在广播的话题上。

用法：

```
rostopic pub [topic] [msg_type] [args]

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

以上命令会发送一条消息给 turtlesim，告诉它以 2.0 大小的线速度和 1.8 大小的角速度开始移动。

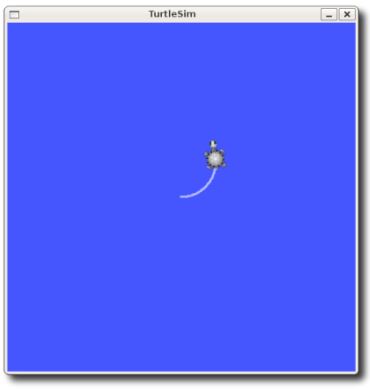
- 

图 6：指令控制 turtle

这是一个非常复杂的例子，因此让我们来详细分析一下其中的每一个参数。

- rostopic pub
    - 这条命令将会发布消息到某个给定的话题。
- -1
    - （单个破折号）这个参数选项使 rostopic 发布一条消息后马上退出。
- /turtle1/cmd_vel
    - 这是消息所发布到的话题名称。
- geometry_msgs/Twist
    - 这是所发布消息的类型。
- --
- （双破折号）这会告诉命令选项解析器接下来的参数部分**都不是**命令选项。这在参数里面包含有破折号-（比如负号）时是必须要添加的。
- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
- 你可能已经注意到 turtle 已经停止移动了。这是因为 turtle 需要一个稳定的频率为 1Hz 的命令流来保持移动状态。我们可以使用 rostopic pub -r 命令来发布一个稳定的命令流：

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

这条命令以 1Hz 的频率发布速度命令到速度话题上。

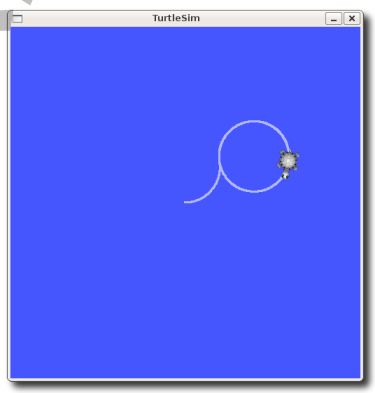
- 

图 7：连续发布指令

我们也可以看一下 rqt_graph 中的情形，可以看到 rostopic 发布器节点（红色）正在与 rostopic echo 节点（绿色）进行通信：

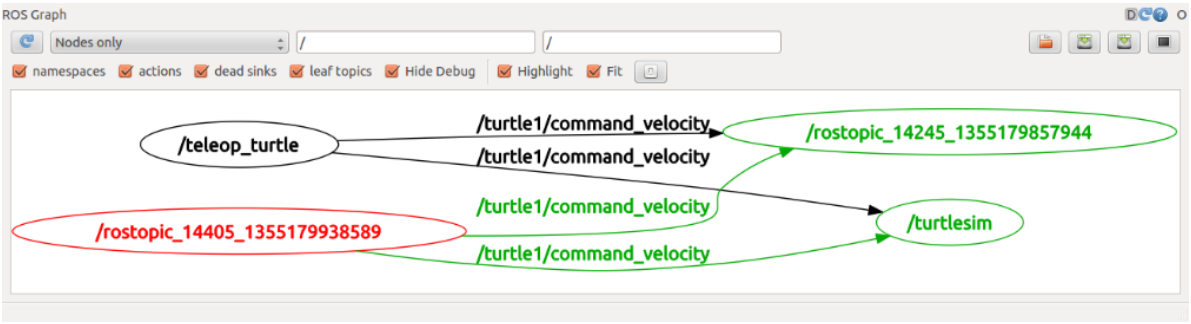


图 8：ROS 节点话题关系图

正如你所看到的，turtle 正沿着一个圆形轨迹连续运动。我们可以在**一个新终端**中通过 rostopic echo 命令来查看 turtlesim 所发布的数据。

使用 rostopic hz

rostopic hz 命令可以用来查看数据发布的频率。

用法：

```
rostopic hz [topic]
```

我们看一下 turtlesim_node 发布/turtle/pose 时有多快：

```
$ rostopic hz /turtle1/pose
```

你会看到：

- subscribed to [/turtle1/pose]
- average rate: 59.354
- min: 0.005s max: 0.027s std dev: 0.00284s window: 58
- average rate: 59.459
- min: 0.005s max: 0.027s std dev: 0.00271s window: 118
- average rate: 59.539
- min: 0.004s max: 0.030s std dev: 0.00339s window: 177
- average rate: 59.492
- min: 0.004s max: 0.030s std dev: 0.00380s window: 237
- average rate: 59.463
- min: 0.004s max: 0.030s std dev: 0.00380s window: 290

现在我们可以知道了 turtlesim 正以大约 60Hz 的频率发布数据给 turtle。我们也可以结合 rostopic type 和 rosmsg show 命令来获取关于某个话题的更深层次的信息：

```
$ rostopic type /turtle1/cmd_vel | rosmsg show
```

到此我们已经完成了通过 rostopic 来查看话题相关情况的过程，接下来我将使用另一个工具来查看 turtlesim 发布的数据。

使用 rqt_plot

rqt_plot 命令可以实时显示一个发布到某个话题上的数据变化图形。这里我们将使用 rqt_plot 命令来绘制正在发布到/turtle1/pose 话题上的数据变化图形。首先，在一个新终端中运行 rqt_plot 命令：

```
$ rosrun rqt_plot rqt_plot
```

或者:

```
$ rqt_plot
```

这会弹出一个新窗口，在窗口左上角的一个文本框里面你可以添加需要绘制的话题。在里面输入/turtle1/pose/x 后之前处于禁用状态的加号按钮将会被使能变亮。按一下该按钮，并对/turtle1/pose/y 重复相同的过程。现在你会在图形中看到 turtle 的 x-y 位置坐标图。

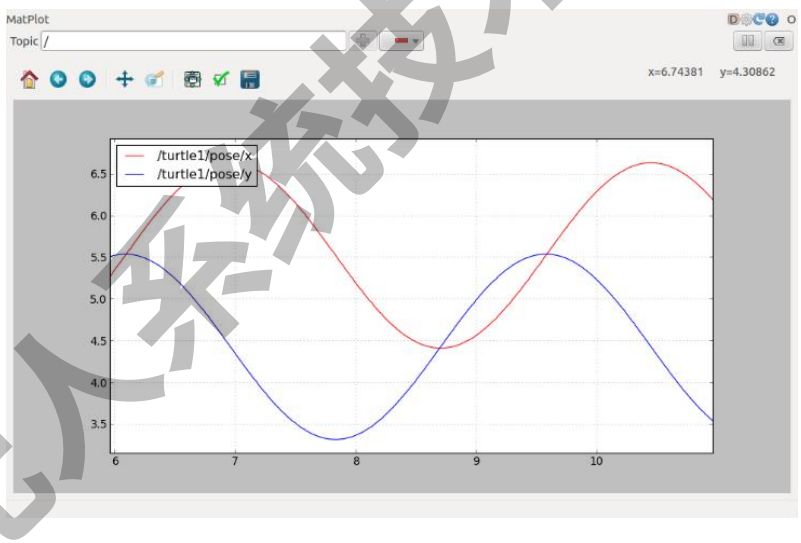


图 9：话题中消息变量的图像化显示

按下减号按钮会显示一组菜单让你隐藏图形中指定的话题。现在隐藏掉你刚才添加的话题并添加/turtle1/pose/theta，你会看到如下图所示的图形：

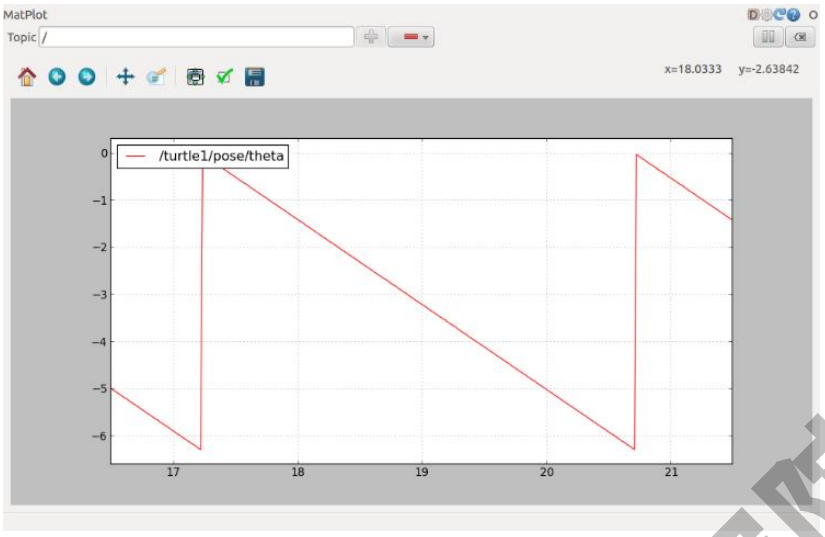


图 10：/turtle1/pose/theta 变量显示