通过练习 turtlesim 认识 ROS

【ROS Turtlesim 官方教程修正版】

内容

Roscore

此命令启动 ROS 并产生 ROS Master，此后 ROS 节点即可登记话题并相互通讯。

$ roscore [1]

参见 ROS wiki 教程 http://wiki.ros.org/roscore

> roscore 是一个基本节点和程序集合，它是基于 ROS 的系统可以运行的先决条件。用户必须确认 ROS master 正在运行中，方可运行任何其他 ROS 节点。ROS master 可由 roscore 命令，或 roslaunch 命令启动。
>
> roscore 将启动:
>
> ROS Master
>
> ROS 参数服务器
>
> rosout 显示节点

运行 roscore 后，可最小化终端，但请保留此中断不被关闭以保证 ROS master 持续运行中。

采用 Turtlesim 学习 ROS 节点, 话题, 和服务

 如果你初学 ROS – 必须耐心。有很多需要学习的资料，而 Turtlesim 则是比较好的起步素材。

此链接为 Turtlesim ROS 官方教程: **http://wiki.ros.org/turtlesim/Tutorials**

其他有用的 ROS 教程如下：

ROS/Tutorials/UnderstandingNodes

ROS/Tutorials/UnderstandingTopics

ROS/Tutorials/UnderstandingServicesParams

Turtlesim Node

让我们启动 turtlesim 节点并探索其特点. 首先在一个终端运行 roscore， 然后在一个新的终端运行 turtlesim 功能包内的 turtlesim 节点：

$ roscore

$ **rosrun turtlesim turtlesim_node** **[2]**

[INFO] [1516751529.792931813]: Starting turtlesim with node name /turtlesim
[INFO] [1516751529.797525686]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

rosrun 命令取这二个参数：[package name] [node name]. 节点生成屏幕图像及乌龟机器人。乌龟处于屏幕中心 x=5.5, y=5.5 。

让我们先了解节点的特点，如 turtlesim 功能包中的话题，服务和消息再设法移动乌龟。

观察 Turtlesim 功能包中的 ROS 节点

rosnode list **[3]**

$ **rosnode list**
/rosout
/turtlesim

注意节点 **/turtlesim** 与 *功能包* **turtlesim** 的异同。

tutor@ubuntu:~$ **rosnode info /turtlesim** **[4]**

```
--------------------------------------------------------------------------------
Node [/turtlesim]
Publications:                (该信息传送至监听 /turtlesim 话题的节点)
 * /turtle1/color_sensor [turtlesim/Color]   (turtlesim 功能包中的颜色消息)

 * /rosout [rosgraph_msgs/Log]
 * /turtle1/pose [turtlesim/Pose](turtlesim 功能包中 /turtle1 的位姿消息)

Subscriptions:
 * /turtle1/cmd_vel [unknown type] (此节点监听速度命令) (用户可用 ROS 服务来操作

乌龟并执行其他操作)

Services: (服务操作格式是 $rosservice call <service> <arguments>)
* /turtle1/teleport_absolute
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level
 * /reset
 * /spawn
 * /clear
 * /turtle1/set_pen
 * /turtle1/teleport_relative
 * /kill

contacting node http://ubuntu:46109/ ...
Pid: 21992
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
```

节点 **/turtlesim** 发布三个话题并接收 **/turtle1/cmd_vel** 话题。节点的服务也一并列出。

移动乌龟的 ROS 服务

```
Services:  (可使用 ROS 服务来操控乌龟以及执行其他操作
- 运行格式是：  $rosservice call <service> <arguments>)
 * /turtle1/teleport_absolute
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level
 * /reset
 * /spawn
 * /clear
```

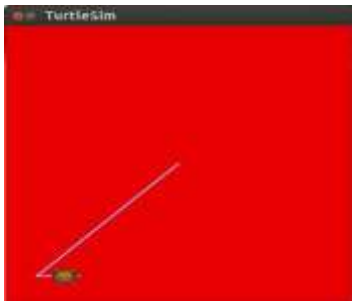
* /turtle1/set_pen
* **/turtle1/teleport_relative**
* /kill

小乌龟可用 rosservice teleport 选项进行移动. 乌龟位姿的格式是 [x y theta].

teleport_absolute

tutor@ubuntu:~/$ **rosservice call /turtle1/teleport_absolute 1 1 0** **[5]**

绝对位移后的乌龟位姿



此后，再相对位移后的乌龟位姿

相对 teleport 选项相对于乌龟的当前位姿来移动乌龟，移动参数是： [linear, angle]

teleport_relative

rosservice call /turtle1/teleport_relative 1 0 **[6]**

现在，乌龟位于 at x=2, y=1（见上图右）.

Turtlesim 节点的话题 Pose

乌龟节点的另一个有用话题是它的位姿 **pose.** 此话题包含坐标 x, y 位置, 乌龟的角度方位, 以及线速度和角速度。

$ rostopic info /turtle1/pose **[7]**

```
Type: turtlesim/Pose
Publishers:
 * /turtlesim (http://D104-45931:42032/)
Subscribers: None
```

此为信息（message）种类

tutor@ubuntu:~$ rostopic type /turtle1/pose **[8]**

```
turtlesim/Pose
```

验证了消息种类

tutor@ubuntu:~$ **rosmsg show turtlesim/Pose** **[9]**

```
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity
```

显示此消息（message）的数据结构

tlharmanphd@D125-43873:/$ **rostopic echo /turtle1/pose** **[10]**

```
x: 2.0
y: 1.0
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 2.0
y: 1.0
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
.
.
.
```

将正在运行的消息结果反映到屏幕终端

屏幕会连续显示当前的位置、方位和速度。可参看屏幕上乌龟的运动，按 Ctrl+c 终止显示。可选择性参考如下 ROS Wiki。

http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

命令行驱动乌龟按圆圈运动 **rostopic pub <command>**

tutor@ubuntu:~$ rosnode info /turtlesim **[11]**

```
--------------------------------------------------------------------------------
Node [/turtlesim]
Publications:
 * /turtle1/color_sensor [turtlesim/Color]
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /turtle1/cmd_vel [此处看不出话题的消息类型]

Services:
 * /turtle1/teleport_absolute
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level
 * /reset
 * /spawn
 * /clear
 * /turtle1/set_pen
 * /turtle1/teleport_relative
 * /kill


contacting node http://D104-45931:42032/ ...
Pid: 4911
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
```

查看 **cmd_vel** 话题消息类型

tutor@ubuntu:~$ rostopic type /turtle1/cmd_vel **[12]**

```
geometry_msgs/Twist
```

再次强调： **rostopic type <topic name>** 显示话题的消息类型

tutor@ubuntu:~$ rosmsg show geometry_msgs/Twist 查看此类型的数据结构 **[13]**

```
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z

geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

用 Linux shell 用户可以同时执行二条指令。 | 操作符用来在 Linux shell 中顺序执行指令。
通过 ｜ 操作符，Linux 系统可将上一条指令的输出作为下一条指令的输入使用 。
操作语法为: 命令行_1 | 命令行_2 | 命令行_3 . . .

$ **rostopic type /turtle1/cmd_vel | rosmsg show** **[14]**

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

话题/turtle1/cmd_vel 的消息种类是 geometry_msgs/Twist， 该消息的数据结构由二个矢量构成，每个矢量含三个元素。

以下链接列出了 ROS 一个基本消息类 **geometry_msgs** 中定义的所有消息种类
http://wiki.ros.org/geometry_msgs

以下命令行显示了当前正在运行的话题及其消息种类：
$ **rostopic list -v** **[15]**

```
Published topics:
 * /turtle1/color_sensor [turtlesim/Color] 1 publisher
 * /rosout [rosgraph_msgs/Log] 1 publisher
 * /rosout_agg [rosgraph_msgs/Log] 1 publisher
 * /turtle1/pose [turtlesim/Pose] 1 publisher

Subscribed topics:
 * /turtle1/cmd_vel [geometry_msgs/Twist] 1 subscriber
* /rosout [rosgraph_msgs/Log] 1 subscriber
```

采用命令行移动乌龟一次

下面的命令行向 turtlesim 节点发送一条消息，告知它以线速度 2.0，角速度 1.8 运动。如此乌龟将沿一个圆周运动后停止。

$ **rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'** **[16]**

-r RATE, --rate=RATE 消息发布频率 (hz).

-1, --once 发布一次消息后退出

小诀窍: 可用 TAB 键补全命令行如下:

尝试练习:

$ rosto (Tab) pub -1 /tur (Tab) cm (Tab) geo (Tab) (Tab) (Tab) ….. [17] With result:

tutor@ubuntu:~$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist "linear:

x: 0.0

y: 0.0

z: 0.0

angular:

x: 0.0

y: 0.0

z: 0.0"

现在可回去输入相应的数值 z= 1.8 and x=0.0. (未执行)

现在可查看，运行命令后乌龟的位姿?

$ rostopic echo /turtle1/pose [18]

x: 3.0583717823

y: 2.39454507828

theta: 1.81439995766

linear_velocity: 0.0

angular_velocity: 0.0

用 CNTL+c 来终止以上输出。

消息 geometry_msgs/Twist 含有二个带三元素的矢量

分别为: 线性与角度 。 例如, '[2.0, 0.0, 0.0]' 是线性空间值 x=2.0, y=0.0, and z=0.0, '[0.0, 0.0, 1.8]' 是沿空间三轴的旋转速度值 x=0.0, y=0.0, and z=1.8. 这些数值为 YAML 格式, 详见: wiki.ros.org/ROS/YAMLCommandLine。

'[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'

上面的例子，你会发现乌龟跑了一小会后会停下来。这是因为乌龟必须稳定地连续接收命令才能驱使它运动，例如以 1Hz 的频率连续发布命令。可用如下指令:

rostopic pub -r 1 command

以下命令显示接收情况:

$ **rosnode info /turtlesim**

Subscribed topics:

* /turtle1/cmd_vel [geometry_msgs/Twist] 1 subscriber **rostopic pub**

驱动乌龟连续沿圆周运动

首先复位乌龟：

tutor@ubuntu:~$ rosservice call /reset **[19]**

tutor@ubuntu:~$ rostopic pub -r 1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]' **[20]**

乌龟沿圆周连续运动

rostopic hz

显示发布消息的频率（单位 Hz），　Crtl-C 退出:

rostopic hz /turtle1/pose **[21]**

```
subscribed to [/turtle1/pose]
average rate: 62.501
        min: 0.016s max: 0.016s std dev: 0.00014s window: 62
average rate: 62.501
        min: 0.016s max: 0.016s std dev: 0.00014s window: 124
average rate: 62.504
        min: 0.016s max: 0.016s std dev: 0.00014s window: 187
average rate: 62.500
        min: 0.016s max: 0.016s std dev: 0.00014s window: 249
average rate: 62.496
        min: 0.015s max: 0.017s std dev: 0.00014s window: 300
```

输出频率大约 60 Hz，即每 16 ms 刷新一次.

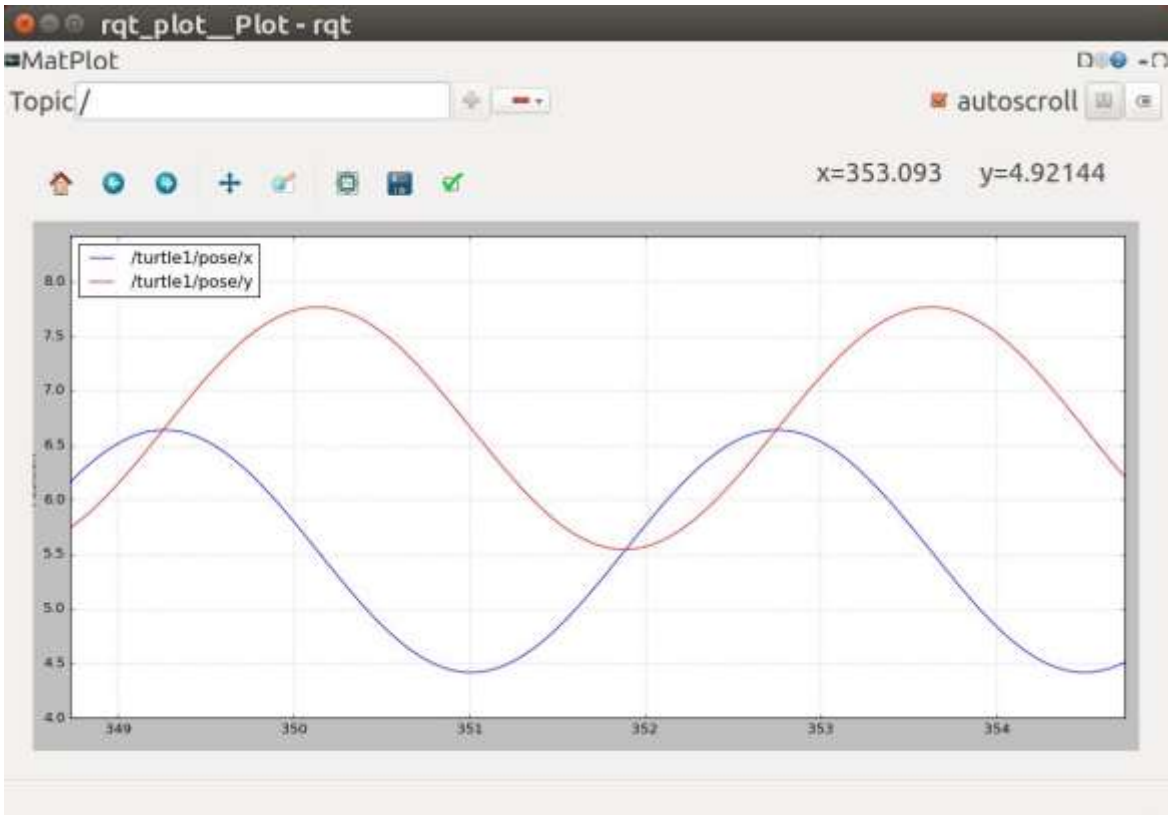
采用 rqt plot 观察 Turtlesim

http://wiki.ros.org/rqt_plot

rqt_plot
可采用此命令打印乌龟的节点和话题信息。

$ rqt_plot /turtle1/pose/x:y:z **[22]**

乌龟在大约 Y 轴 5.5~8，X 轴 4.5 to 6.5 范围内运动。



更多打印配置，详见如下链接：

http://wiki.ros.org/rqt_plot

如下二个命令行指令执行同样的操作：

```
$ rqt_plot /turtle1/pose/x:y:z
$ rqt_plot /turtle1/pose/x /turtle1/pose/y /turtle1/pose/z
```

如果用户希望改变打印的话题，则必须重启命令行指令并给出相应的话题名称。

键盘控制

在 roscore 和 turtlesim_node 已经在二个窗口终端中运行的情况下，重新打开一个终端，运行一个键盘操控乌龟的节点。

$ rosrun turtlesim turtle_teleop_key **[23]**

```
tutor@ubuntu:~$ rosrun turtlesim turtle_teleop_key
        Reading from keyboard
        ---------------------------
        Use arrow keys to move the turtle.
        Up arrow       Turtle In Turtle's x direction
        Down arrow     Turtle In Turtles's -x direction
        Right arrow    Rotate CW
        Left  arrow    Rotate CCW
```

tutor@ubuntu:~$ rosnode list **[24]**

```
        /rosout
        /teleop_turtle
        /turtlesim
```

注意，现在我们看到了刚运行的名为 /teleop_turtle 的节点

tutor@ubuntu:~$ rosnode info /teleop_turtle **[25]**

```
--------------------------------------------------------------------------------
        Node [/teleop_turtle]
        Publications:
         * /turtle1/cmd_vel [geometry_msgs/Twist]
         * /rosout [rosgraph_msgs/Log]

        Subscriptions: None

        Services:
         * /teleop_turtle/get_loggers
         * /teleop_turtle/set_logger_level

        contacting node http://D104-45931:43692/ ...
        Pid: 8381
        Connections:
         * topic: /rosout
           * to: /rosout
           * direction: outbound
           * transport: TCPROS
         * topic: /turtle1/cmd_vel
           * to: /turtlesim
           * direction: outbound
        * transport: TCPROS
```

/teleop_turtle 节点正发布话题 /turtle1/cmd_vel
你可以辨认出此话题的消息种类吗？

此处显示 /teleop_turtle 没有订阅任何话题

此处可见话题 **/turtle1/cmd_vel 在发布中 [消息种类为 geometry_msgs/Twist]**

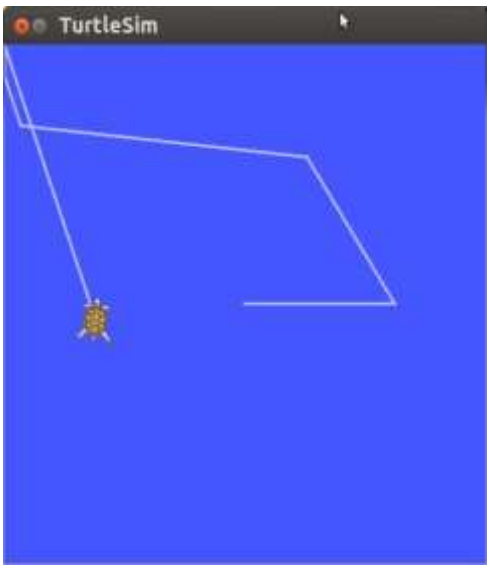
此时，在运行了/teleop_turtle 节点后，我们可以回顾观察 /turtlesim 节点

tutor@ubuntu:~$ rosnode info /turtlesim **[26]**

```
--------------------------------------------------------------------------------
Node [/turtlesim]
Publications:
 * /turtle1/color_sensor [turtlesim/Color]
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /turtle1/cmd_vel [geometry_msgs/Twist]

Services:
 * /turtle1/teleport_absolute
 * /reset
 * /clear
 * /turtle1/teleport_relative
 * /kill
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level
 * /spawn
 * /turtle1/set_pen

contacting node http://ubuntu:46109/ ...
Pid: 7956
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /turtle1/pose
    * to: /rqt_gui_py_node_22321
    * direction: outbound
    * transport: TCPROS
          * topic: /turtle1/cmd_vel
            * to: /teleop_turtle (http://ubuntu:43929/)
            * direction: inbound
* transport: TCPROS
```

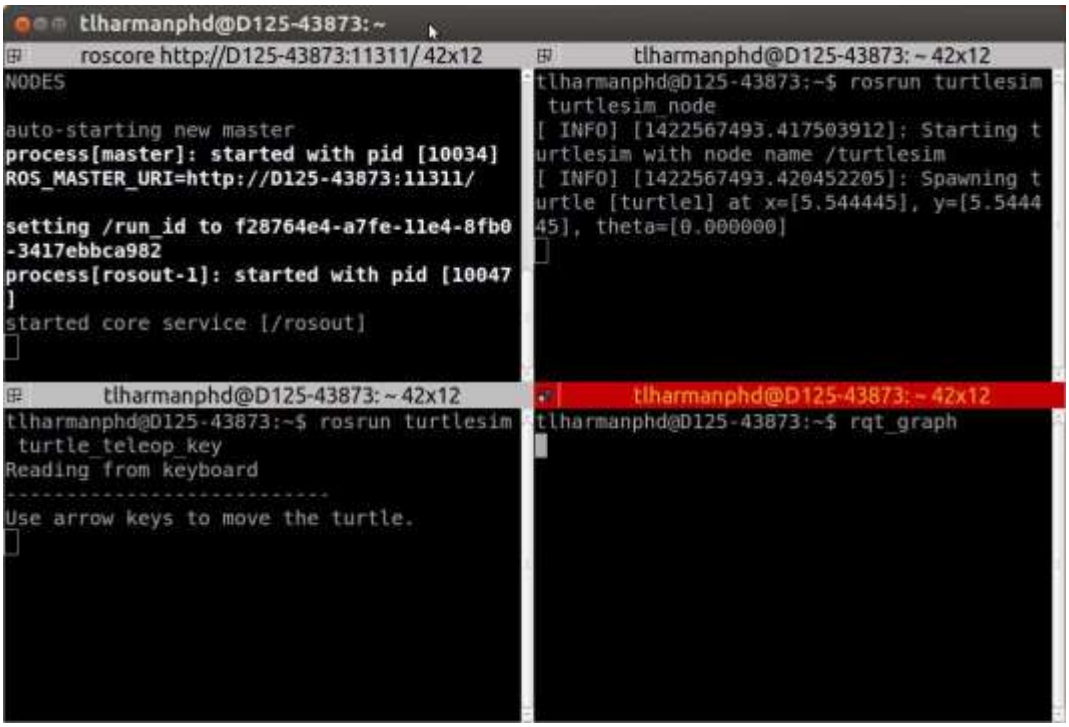
注意到此处: 节点 **/teleop_turtle** 发布新话题 **/turtle1/cmd_vel** 至节点/turtlesim，也可通过 rqt_graph 直观看到。

使用键盘上的上下左右箭头按钮移动乌龟，注意屏幕的焦点需要是运行 **turtle_teleop_key** 的终端，否则乌龟不会移动。



Turtlesim keyboard control

现在用户可以再打开第四个终端，运行 rqt_graph 来观察节点和消息。四个窗口及其命令行见下图。



(Publications and Subscriptions)

tutor@ubuntu:~$ rostopic list **[27]**

```
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

此处/turtle1/cmd_vel 为一个比较重要的话题，可用键盘，或者使用 rostopic pub 来发布此话题。

查看话题 /turtle1/cmd_vel 的数据结构

rostopic echo 命令显示节点发送的控制乌龟的数据结构。当移动乌龟时，数据按发送频率更新。当使用键盘移动乌龟时，实时的速度数据会在屏幕显示: 如线性移动则显示 x 方向数值，如旋转则显示 沿 z 轴的角速度 。

tutor@ubuntu:~$ **rostopic echo /turtle1/cmd_vel** **[28]**

```
linear:
  x: 2.0          (向前线速度)
  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
---
linear:
  x: -2.0
  y: 0.0
```

```
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 2.0(沿 Z 轴的反时针角速度)
---
.
.
```

上面显示了 **cmd_vel** 话题的参数，为线速度和角速度。此处乌龟一直直线运动，直到最后发生了旋转。

为了找到乌龟在背景蓝色海洋中的位姿，可以使用话题 **/turtle1/pose**

tutor@ubuntu:~$ **rostopic echo /turtle1/pose** **[29]**

```
x: 5.544444561
y: 5.544444561
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
.
.
```

按 CNTL+c 终止显示输出。

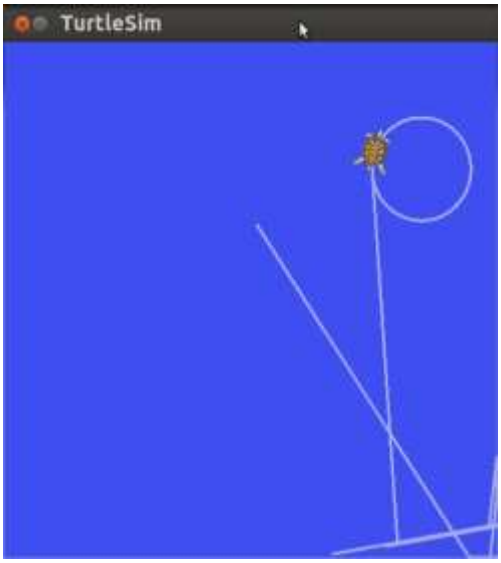
若用户返回键控窗口 teleop_key，使用箭头来移动乌龟，则用户会看到位姿消息 (turtlesim/Pose) 发生变化。记住使用如下命令查看消息 turtlesim/Pose 数据结构：

tutor@ubuntu:~$ **rosmsg show turtlesim/Pose** float32 x

```
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity
```

也可发布话题 **turtle1/cmd_velocity** 使得乌龟沿着圆周运行。

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



乌龟对发布的话题的反应

此处命令以 1Hz 频率发布话题发布，发布的话题/turtle1/command_velocity 后面需紧跟消息种类 turtlesim/Velocity，使得乌龟以线速度 2.0，角速度 1.8 沿圆周运动。 详见官网教程:

http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

如前所述，消息 turtlesim/Velocity 具有二个浮点元素 : `linear` 和 `angular`. 此处, `2.0` 是线速度值, `1.8` 是角速度值.

tutor@ubuntu:~$ **rosmsg show turtlesim/Velocity**

```
float32 linear
float32 angular
```

清除 turtlesim 屏幕内容:

tutor@ubuntu:~$ rosservice call /clear