ROS 常用功能包安装及常用命令行工具

一、常用工具包：以下功能包供参考，除了 Git 工具外，建议在需要时再安装。

1. ROS Control 功能包栈：一组包含控制器接口 API、控制器管理软件、硬件接口和数据传输的功能包（更多信息参见 http://wiki.ros.org/ros_control）。

```
% sudo apt-get install ros-melodic-ros-control ros-melodic-ros-controllers
% sudo apt-get install ros-melodic-rqt-controller-manager ros-melodic-rqt-joint-trajectory-controller
% sudo apt-get install ros-melodic-ddynamic-reconfigure
% sudo apt-get install ros-melodic-gazebo-ros-pkgs ros-melodic-gazebo-ros-control
```

2. ROS tf 教程功能包：tf 是一个可让用户随时间动态跟踪各个坐标系间关系的功能包(更多信息见 http://wiki.ros.org/tf/Tutorials/Introduction%20to%20tf)

```
% sudo apt-get install ros-melodic-ros-tutorials ros-melodic-geometry-tutorials ros-melodic-rviz ros-melodic-rosbash ros-melodic-rqt-tf-tree
```

3. ROS 关节状态发布 GUI 功能包：采用滑块图形化控制机器人各个关节。

```
% sudo apt install ros-melodic-joint-state-publisher-gui
```

4. Git 工具：分布式软件版本控制器，从第三方网站，如 github 等，下载软件的工具。

```
% sudo apt-get install git
```

二、ROS 常用命令行：

1. roscore

roscore 是运行 ROS 系统前必须先运行的命令，它启动 ROS 服务器。

- 在使用 ROS 时，应当始终让 roscore 运行；
- 用户通常先专门打开一个终端运行 roscore，然后再在其他终端上运行 ROS 程序。
- 除非用户完成了 ROS 相关工作，否则不必中断 roscore。
- 当用户已经完成 ROS 工作需要退出 ROS 服务，则在先前运行 roscore 的终端里点击 *Ctrl-C* 组合键退出 ROS 环境。

重要提示：ROS 节点程序创建后会和 ROS 服务器建立联系，此为一次性动作，各 ROS 节点此后不会再次尝试和 ROS 服务器建立联系，如果因 roscore 启动的服务器被终止（即 roscore 被中断），则所有和此服务器建立联系的 ROS 节点程序也会永久停止运行。重启服务器后，这些节点程序也需要重新启动。

2. rosrun

可以在不知某一个功能包路径的情况下，运行该功能包内的 ROS 节点。

```
% rosrun [package_name] [executable_name]
```

小练习：
打开三个屏幕终端，在不同终端分别运行下面三条指令：

```
% roscore
% sudo apt-get install ros-$(rosversion -d)-turtlesim
% rosrun turtlesim turtlesim_node
% rosrun turtlesim turtle_teleop_key
```

用户会看到一个乌龟机器人在它的空间里受控行动。

3. 获取 ROS 系统信息

3.1 rosnode

rosnode 显示当前正在运行的 ROS 节点信息。

```
rosnode info    print information about node
rosnode kill    kill a running node
rosnode list    list active nodes
rosnode machine list nodes running on a particular machine or list machines
rosnode ping    test connectivity to node
rosnode cleanup purge registration information of unreachable nodes
```

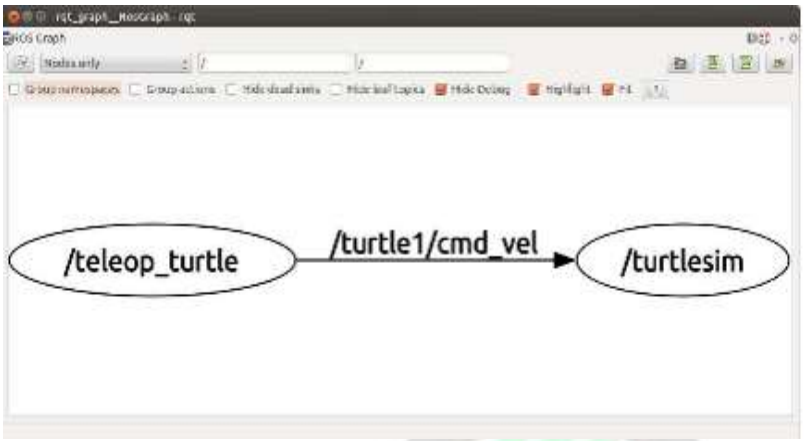
可针对上面的小练习尝试运行 rosnode list。

3.2 rqt_graph

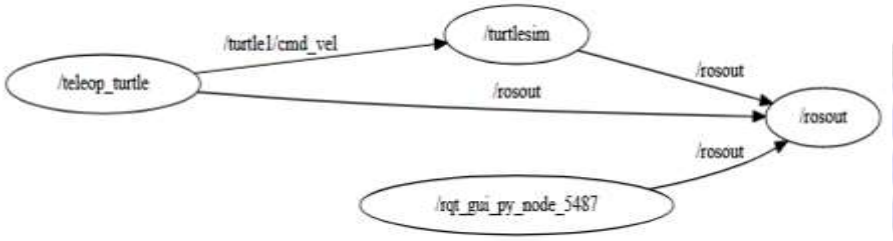
rqt_graph 是观察 ROS 节点之间信息交互的最直观有效工具。
以上面的乌龟机器人为例，在新开的终端运行以下指令。

```
% rqt_graph
```

用户将会看到二个节点之间通过一个消息传递控制信息，如下图示。



如果不选择“Debug”选项，则可看到全部节点如下。



提示：此处 rqt_graph 本身也是一个节点。所有三个节点向 ROS 输出节点/rosout 发布话题。/rosout 本身既是节点，也是话题，ROS 服务器不会混淆/rosout。

3.3 rostopic

rostopic 可用来获得 ROS 话题的信息（如数据速率、实际数据、发布/接收节点等）。

```
$ rostopic -h
```

```
rostopic bw     display bandwidth used by topic
rostopic delay display delay for topic which has header
rostopic echo   print messages to screen
rostopic find   find topics by type
rostopic hz     display publishing rate of topic
rostopic info   print information about active topic
rostopic list   print information about active topics
rostopic pub    publish data to topic
rostopic type   print topic type
```

小练习：

1. 用 rostopic 列出本例所有的话题，检测/teleop_turtle 节点发布话题的速率，以及当用键盘操控乌龟机器人时，话题占用的带宽。
2. 观察话题 /turtle1/cmd_vel，当键盘操控机器人时，它的变化。
3. 查看话题 /turtle1/cmd_vel 的种类。

3.4 rosmsg

通过 rosmsg 可以看到消息种类的数据结构。

```
$ 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
```

小练习：

使用 rosmsg -h 了解所有 rosmsg 的命令行选项。列出所有消息，分析 /turtlesim/Pose 的消息数据结构。

参考资料：

http://wiki.ros.org/turtlesim

英佳 科技