Understanding ROS Nodes

**Description:** This tutorial introduces ROS graph concepts and discusses the use of [roscore](http://wiki.ros.org/roscore), [rosnode](http://wiki.ros.org/rosnode), and [rosrun](http://wiki.ros.org/rosrun)commandline tools.  
  
**Tutorial Level:** BEGINNER  
  
**Next Tutorial:**  [Understanding ROS topics](http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics)

**Contents**

1. [Prerequisites](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes#Prerequisites)
2. [Quick Overview of Graph Concepts](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes#Quick_Overview_of_Graph_Concepts)
3. [Nodes](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes#Nodes)
4. [Client Libraries](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes#Client_Libraries)
5. [roscore](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes#roscore)
6. [Using rosnode](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes#Using_rosnode)
7. [Using rosrun](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes#Using_rosrun)
8. [Review](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes#Review)

Prerequisites

For this tutorial we'll use a lighweight simulator, to install it run the following command:

$ sudo apt-get install ros-<distro>-ros-tutorials

Replace '<distro>' with the name of your ROS distribution (e.g. indigo, jade, kinetic)

Quick Overview of Graph Concepts

* [Nodes](http://wiki.ros.org/Nodes): A node is an executable that uses ROS to communicate with other nodes.
* [Messages](http://wiki.ros.org/Messages): ROS data type used when subscribing or publishing to a topic.
* [Topics](http://wiki.ros.org/Topics): Nodes can *publish* messages to a topic as well as *subscribe* to a topic to receive messages.
* [Master](http://wiki.ros.org/Master): Name service for ROS (i.e. helps nodes find each other)
* [rosout](http://wiki.ros.org/rosout): ROS equivalent of stdout/stderr
* [roscore](http://wiki.ros.org/roscore): Master + rosout + parameter server (parameter server will be introduced later)

Nodes

A node really isn't much more than an executable file within a ROS package. ROS nodes use a ROS client library to communicate with other nodes. Nodes can publish or subscribe to a Topic. Nodes can also provide or use a Service.

Client Libraries

ROS client libraries allow nodes written in different programming languages to communicate:

* rospy = python client library
* roscpp = c++ client library

roscore

roscore is the first thing you should run when using ROS.

Please run:

$ roscore

You will see something similar to:

* ... logging to ~/.ros/log/9cf88ce4-b14d-11df-8a75-00251148e8cf/roslaunch-machine\_name-13039.log
* Checking log directory for disk usage. This may take awhile.
* Press Ctrl-C to interrupt
* Done checking log file disk usage. Usage is <1GB.
* started roslaunch server http://machine\_name:33919/
* ros\_comm version 1.4.7
* SUMMARY
* ======
* PARAMETERS
* \* /rosversion
* \* /rosdistro
* NODES
* auto-starting new master
* process[master]: started with pid [13054]
* ROS\_MASTER\_URI=http://machine\_name:11311/
* setting /run\_id to 9cf88ce4-b14d-11df-8a75-00251148e8cf
* process[rosout-1]: started with pid [13067]
* started core service [/rosout]

If roscore does not initialize, you probably have a network configuration issue. See [Network Setup - Single Machine Configuration](http://www.ros.org/wiki/ROS/NetworkSetup#Single_machine_configuration)

If roscore does not initialize and sends a message about lack of permissions, probably the ~/.ros folder is owned by root, change recursively the ownership of that folder with:

$ sudo chown -R <your\_username> ~/.ros

Using rosnode

Open up a **new terminal**, and let's use **rosnode** to see what running roscore did... Bear in mind to keep the previous terminal open either by opening a new tab or simply minimizing it.

**Note:** When opening a new terminal your environment is reset and your ~/.bashrc file is sourced. If you have trouble running commands like rosnode then you might need to add some environment setup files to your ~/.bashrc or manually re-source them.

rosnode displays information about the ROS nodes that are currently running. The rosnode list command lists these active nodes:

$ rosnode list

* You will see:
* /rosout

This showed us that there is only one node running: [rosout](http://wiki.ros.org/rosout). This is always running as it collects and logs nodes' debugging output.

The rosnode info command returns information about a specific node.

$ rosnode info /rosout

This gave us some more information about rosout, such as the fact that it publishes /rosout\_agg.

* ------------------------------------------------------------------------
* Node [/rosout]
* Publications:
* \* /rosout\_agg [rosgraph\_msgs/Log]
* Subscriptions:
* \* /rosout [unknown type]
* Services:
* \* /rosout/get\_loggers
* \* /rosout/set\_logger\_level
* contacting node http://machine\_name:54614/ ...
* Pid: 5092

Now, let's see some more nodes. For this, we're going to use rosrun to bring up another node.

Using rosrun

rosrun allows you to use the package name to directly run a node within a package (without having to know the package path).

Usage:

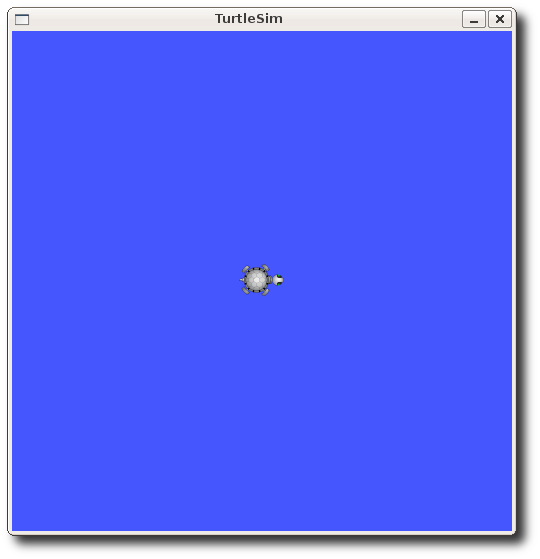
$ rosrun [package\_name] [node\_name]

So now we can run the turtlesim\_node in the turtlesim package.

Then, in a **new terminal**:

$ rosrun turtlesim turtlesim\_node

You will see the turtlesim window:

* 

**NOTE:** The turtle may look different in your turtlesim window. Don't worry about it - there are [many types of turtle](http://wiki.ros.org/Distributions#Current_Distribution_Releases) and yours is a surprise!

In a **new terminal**:

$ rosnode list

You will see something similar to:

* /rosout
* /turtlesim

One powerful feature of ROS is that you can reassign Names from the command-line.

Close the turtlesim window to stop the node (or go back to the rosrun turtlesim terminal and use ctrl-C). Now let's re-run it, but this time use a [Remapping Argument](http://wiki.ros.org/Remapping%20Arguments) to change the node's name:

$ rosrun turtlesim turtlesim\_node \_\_name:=my\_turtle

Now, if we go back and use rosnode list:

$ rosnode list

* You will see something similar to:
* /my\_turtle
* /rosout

Note: If you still see /turtlesim in the list, it might mean that you stopped the node in the terminal using ctrl-C instead of closing the window, or that you don't have the $ROS\_HOSTNAME environment variable defined as described in [Network Setup - Single Machine Configuration](http://www.ros.org/wiki/ROS/NetworkSetup#Single_machine_configuration). You can try cleaning the rosnode list with:  $ rosnode cleanup

We see our new /my\_turtle node. Let's use another rosnode command, ping, to test that it's up:

$ rosnode ping my\_turtle

* rosnode: node is [/my\_turtle]
* pinging /my\_turtle with a timeout of 3.0s
* xmlrpc reply from http://aqy:42235/ time=1.152992ms
* xmlrpc reply from http://aqy:42235/ time=1.120090ms
* xmlrpc reply from http://aqy:42235/ time=1.700878ms
* xmlrpc reply from http://aqy:42235/ time=1.127958ms

Review

What was covered:

* roscore = ros+core : master (provides name service for ROS) + rosout (stdout/stderr) + parameter server (parameter server will be introduced later)
* rosnode = ros+node : ROS tool to get information about a node.
* rosrun = ros+run : runs a node from a given package.