ROS: Quick Reference

Contents

1 Filesystem concepts

- **package**: software organization unit of ROS code; each package contains libraries, executables, scripts, or other artifacts
- **manifest** (`package.xml`): description of a package – define depedencies between packages and contains meta info (e.g. version, license, maintainer, etc.)
- `rospack`: get info about packges
- `roscd` chdir to roscpp package

2 Basic Concepts

- **node**: an executable that uses to communicate with other nodes
  - not much more than an executable within a ROS package
  - ROS node uses ROS client library to communicate with other nodes
  - nodes can publish or subscribe to a topic
  - nodes can provide or use a service
- **message**: ROS datatype used when subscribing or publishing to a topic
  -
- **topic**: nodes can publish messages to a topic as well as subscribe to a topic to receive message.
  - like **LISTSERV** or **USENET newsgroup**
- **master**: name service for ROS (helps nodes find each other)
- **rosout**: ROS equivalent of stdout/stderr
- **roscore**: Master + rosout + parameter server
  - can be run with `roscore` program

3 Running ROS core

```
# run the core server
$ roscore

# list the ROS nodes
$ rosnode list
/rosout

# query node info
$ rosnode info /rosout
publications:
  * /rosout_agg

subscriptions:
  * /rosout
```

```
services:
  * /rosout/set_logger_level
  * /rosout/get_loggers

# rosrun: use the package name to directly run a node within a package
# (without having to know the package path)
#   rosrun [package_name] [node_name]
$ rosrun turtlesim turtlesim_node

# after this, we have one more node
$ rosnode list
/rosout
/turtlesim
```

4 ROS Topics

```
$ 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

# display a verbose list of topics to publish to and subscribe to
# and their type
$ rostopic list -v
Published topics:
 * /turtle1/color_sensor [turtlesim/Color] 1 publisher
 * /turtle1/command_velocity [turtlesim/Velocity] 1 publisher
 * /rosout [roslib/Log] 2 publishers
 * /rosout_agg [roslib/Log] 1 publisher
 * /turtle1/pose [turtlesim/Pose] 1 publisher

Subscribed topics:
 * /turtle1/command_velocity [turtlesim/Velocity] 1 subscriber
 * /rosout [roslib/Log] 1 subscriber

# 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]'
```

5 ROS Messages

Communication on topics happens by sending ROS messages between nodes. For the publisher (`turtle_teleop_key`) and subscriber (`turtlesim_node`) to communicate, the publisher and subscriber must send and receive the same type of message. This means that a topic type is defined by the message type published on it. The type of the message sent on a topic can be determined using rostopic type.

```
# rostopic type [topic]
$ rostopic type /turtle1/cmd_vel
geometry_msgs/Twist

# look at the details of the message using rosmsg
$ 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
```

6 Creating a ROS msg and srv

- **msg**: msg files are text files that describe the fields of ROS message; used to generate source code for messages in different langauges

7 Writing a Simple Service and Client (C++)

```
#include "ros/ros.h"
#include "beginner_tutorials/AddTwoInts.h"

bool add(beginner_tutorials::AddTwoInts::Request  &req,
         beginner_tutorials::AddTwoInts::Response &res)
{
  res.sum = req.a + req.b;
  ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
  ROS_INFO("sending back response: [%ld]", (long int)res.sum);
  return true;
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "add_two_ints_server");
  ros::NodeHandle n;

  ros::ServiceServer service = n.advertiseService("add_two_ints", add);
  ROS_INFO("Ready to add two ints.");
  ros::spin();

  return 0;
}
```