Chapter 2
WS-Package-Communication

Wende Ke

Department of Mechanical and Energy Engineering

Southern University of Science and Technology

Installing and Configuring Your ROS Environment

$ mkdir -p ~/catkin_ws/src	Create the directorie *catkin_ws* and its sub dir *src*.
$ cd ~/catkin_ws/	Change to the directorie *catkin_ws*.
$ catkin_make	Compile *catkin_ws* and create a *CMakeLists.txt* link in your 'src' folder.
$ source devel/setup.bash	Make sure ROS_PACKAGE_PATH environment variable includes the directory *catkin_ws*.
$ echo $ROS_PACKAGE_PATH	Show the path of ROS_PACKAGE_PATH.

https://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

More links:

https://wiki.ros.org/ROS/EnvironmentVariables

Navigating the ROS Filesystem

$ sudo apt-get install ros-noetic-ros-tutorials	Inspect a package in ros-tutorials.
$ rospack find roscpp	Find the package named *roscpp*.
$ roscd roscpp	Change to the dir *roscpp*.
$ pwd	Show the working directory.
$ roscd log	Show the dir of *log*.
$ rosls roscpp_tutorials	List the dir *roscpp_tutorials*.
$ roscd roscpp_tut<<< now push the TAB key >>>	Tab Completion.

https://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem

More links:

https://wiki.ros.org/rosbash

Creating a ROS Package

```
workspace_folder/        -- WORKSPACE
    src/                 -- SOURCE SPACE
        CMakeLists.txt      -- 'Toplevel' CMake file, provided by catkin
        package_1/
            CMakeLists.txt     -- CMakeLists.txt file for package_1
            package.xml        -- Package manifest for package_1
        ...
        package_n/
            CMakeLists.txt     -- CMakeLists.txt file for package_n
            package.xml        -- Package manifest for package_n
```

https://wiki.ros.org/ROS/Tutorials/CreatingPackage

$ cd ~/catkin_ws/src	Change to the dir *catkin_ws/src*.
$ catkin_create_pkg beginner_tutorials std_msgs rospy roscpp	Create a new package called 'beginner_tutorials' which depends on std_msgs, roscpp, and rospy.
$ cd ~/catkin_ws	Change to the dir *catkin_ws*.
$ catkin_make	Compile the package *beginner_tutorials.*

More links:

https://wiki.ros.org/catkin/package.xml

https://wiki.ros.org/catkin/CMakeLists.txt

Understanding ROS Nodes

$ sudo apt-get install ros-noetic-ros-tutorials	Install the package *ros_tutorials*.
$ roscore	Initialize the network configuration
$ rosnode list	Displays information about the ROS nodes that are currently running.
$ rosnode info /rosout	Returns information about a specific node.
$ rosrun turtlesim turtlesim_node	Run the *turtlesim_node* in the *turtlesim* package.
$ rosrun turtlesim turtle_teleop_key	Run the *turtle_teleop_key* in the *turtlesim* package to operate the turtle.
$ rosnode ping turtlesim	Check the network connection of *turtlesim*.

https://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

More links:

https://wiki.ros.org/roscore

Understanding ROS Topics

$ rosrun turtlesim turtlesim_node	Run the *turtlesim_node* in the *turtlesim* package.
$ rosrun turtlesim turtle_teleop_key	Run the *turtle_teleop_key* in the *turtlesim* package to operate the turtle.
$ sudo apt-get install ros-noetic-rqt	rqt_graph creates a dynamic graph of what's going on in the system.
$ sudo apt-get install ros-noetic-rqt-common-plugins	Install the *common-plugins* of rqt_graph.
$ rosrun rqt_graph rqt_graph	Run the *rqt_graph* in the *rqt_graph* package.
$ rostopic -h	Get information about ROS topics.
$ rostopic echo /turtle1/cmd_vel	The data is published on the */turtle1/cmd_vel* topic.

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

More links:

https://wiki.ros.org/rqt_graph

$ rostopic type /turtle1/cmd_vel	Show the type of message.
$ rosmsg show geometry_msgs/Twist	Check the details of the message.
$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'	send a single message to turtlesim.
$ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, -1.8]'	a steady stream of commands at 1 Hz to keep moving.
$ rosrun rqt_graph rqt_graph	Run the *rqt_graph* in the *rqt_graph* package.
$ rostopic hz /turtle1/pose	How fast the turtlesim_node is publishing /turtle1/pose.
$ rosrun rqt_plot rqt_plot	Displays a scrolling time plot of the data published on topics..

More links:

https://wiki.ros.org/ROS/YAMLCommandLine

Understanding ROS Services and Parameters

$ rosservice list	Shows us that the turtlesim node provides nine services.
$ rosservice type /clear	Find out what type the clear service is.
$ rosservice call /clear	Clears the background of the turtlesim_node.
$ rosservice type /spawn | rossrv show	The service has arguments by looking at the information for the service spawn.
$ rosservice call /spawn 2 2 0.2 ""	spawn a new turtle at a given location and orientation.
$ rosparam list	The turtlesim node has three parameters on the param server for background color.
$ rosparam set /turtlesim/background_r 150	Change the red channel of the background color.

https://wiki.ros.org/ROS/Tutorials/UnderstandingServicesParams

More links:

https://wiki.ros.org/Parameter%20Server

Understanding ROS Services and Parameters

$ rosservice call /clear	Call the clear service for the parameter change to take effect.
$ rosparam get /turtlesim/background_g	Get the value of the green background channel.
$ rosparam get /	Show us the contents of the entire Parameter Server.
$ rosparam dump params.yaml	Write all the parameters to the file params.yaml.
$ rosparam load params.yaml copy_turtle	Load these yaml files into new namespaces, e.g. copy_turtle.
$ rosparam get /copy_turtle/turtlesim/background_b	We will get the number 255.

Using rqt_console and roslaunch

```
$ sudo apt-get install ros-noetic-rqt ros-noetic-rqt-common-plugins ros-noetic-turtlesim
```

Install both packages.

```
$ rosrun rqt_console rqt_console
```

```
$ rosrun rqt_logger_level rqt_logger_level
```

```
$ rosrun turtlesim turtlesim_node
```

```
$ rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
```

```
$ roscd beginner_tutorials
```

```
$ mkdir launch
```

```
$ cd launch
```

https://wiki.ros.org/ROS/Tutorials/UsingRqtconsoleRoslaunch#CA-91a3946a9c4cf7301bb55ec0c3f8a77f6c8f9777_1

Create a launch file called turtlemimic.launch and paste the following:

```
<launch>

 <group ns="turtlesim1">
     <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
 </group>

 <group ns="turtlesim2">
     <node pkg="turtlesim" name="sim" type="turtlesim_node"/>
 </group>

 <node pkg="turtlesim" name="mimic" type="mimic">
     <remap from="input" to="turtlesim1/turtle1"/>
     <remap from="output" to="turtlesim2/turtle1"/>
 </node>

</launch>
```

$ roslaunch beginner_tutorials turtlemimic.launch	roslaunch the launch file.

```
$ rostopic pub /turtlesim1/turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, -1.8]'
```

```
$ rqt_graph
```

More links:

https://wiki.ros.org/rqt_console

https://wiki.ros.org/rqt_logger_level

https://wiki.ros.org/roslaunch

Understanding ROS Services and Parameters

```
$ roscd beginner_tutorials
```

```
$ mkdir msg
```

```
$ echo "int64 num" > msg/Num.msg
```

```
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

Open *package.xml*, and make sure these two lines are in it and uncommented.

```
# Do not just add this to your CMakeLists.txt, modify the existing text to
add message_generation before the closing parenthesis
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
  message_generation
)
```

Open *CMakeLists.txt*, and make sure these modifications.

```
catkin_package(
        ...
        CATKIN_DEPENDS message_runtime ...
        ...)
```

Make sure you export the message runtime dependency.

```
add_message_files(
        FILES
        Num.msg
)
```

Uncomment it by removing the # symbols.

```
generate_messages(
        DEPENDENCIES
        std_msgs
)
```

Uncomment it by removing the # symbols.

```
$ rosmsg show beginner_tutorials/Num
```

You will see: int64 num

https://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv

More links:

https://www.htmlhelp.com/reference/wilbur/misc/comment.html

```
$ roscd beginner_tutorials
```

```
$ mkdir srv
```

```
$ roscp rospy_tutorials AddTwoInts.srv srv/AddTwoInts.srv
```

copy a service from the rospy_tutorials package.

```
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

Open *package.xml*, and make sure these two lines are in it and uncommented.

```
# Do not just add this line to your CMakeLists.txt, modify the existing line
find_package(catkin REQUIRED COMPONENTS
        roscpp
        rospy
        std_msgs
        message_generation
)
```

Open *CMakeLists.txt*, and make sure these modifications.

```
add_service_files(
        FILES
        AddTwoInts.srv
)
```

Open *CMakeLists.txt*, and make sure these modifications.

```
generate_messages(
        DEPENDENCIES
        std_msgs
)
```

Open *CMakeLists.txt*, and make sure these modifications.

```
$ rossrv show beginner_tutorials/AddTwoInts
```

```
$ roscd beginner_tutorials
```

```
$ cd ../..
```

```
$ catkin_make
```

Writing a Simple Publisher and Subscriber(C++)

$ roscd beginner_tutorials

$ mkdir -p src

Create the *src/talker.cpp* file within the beginner_tutorials package and paste the following inside it:
https://raw.github.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/talker/talker.cpp

https://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29

More links:

http://docs.ros.org/en/api/std_msgs/html/msg/String.html

https://wiki.ros.org/Names#Graph

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <sstream>

int main(int argc, char **argv)
{
  ros::init(argc, argv, "talker");
  ros::NodeHandle n;
  ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);
  ros::Rate loop_rate(10);
  int count = 0;
  while (ros::ok())
  {
    std_msgs::String msg;
    std::stringstream ss;
    ss << "hello world " << count;
    msg.data = ss.str();
    ROS_INFO("%s", msg.data.c_str());
    chatter_pub.publish(msg);
    ros::spinOnce();
    loop_rate.sleep();
    ++count;
  }
  return 0;
}
```

Writing a Simple Publisher and Subscriber(C++)

Create the *src/listener.cpp* file within the beginner_tutorials package and paste the following inside it:
https://raw.github.com/ros/ros_tutorials/kinetic-devel/roscpp_tutorials/listener/listener.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"

void chatterCallback(const std_msgs::String::ConstPtr& msg)
{
  ROS_INFO("I heard: [%s]", msg->data.c_str());
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "listener");
  ros::NodeHandle n;
  ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);
  ros::spin();

  return 0;
}
```

Simply add these few lines to the bottom of your *CMakeLists.txt*:

```
add_executable(talker src/talker.cpp)
target_link_libraries(talker ${catkin_LIBRARIES})
add_dependencies(talker beginner_tutorials_generate_messages_cpp)

add_executable(listener src/listener.cpp)
target_link_libraries(listener ${catkin_LIBRARIES})
add_dependencies(listener beginner_tutorials_generate_messages_cpp)
```

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

```
$ source ./devel/setup.bash
```

```
$ rosrun beginner_tutorials talker
```

```
$ rosrun beginner_tutorials listener
```

Writing a Simple Service and Client (C++)

$ roscd beginner_tutorials

Create the *src/add_two_ints_server.cpp* file within the beginner_tutorials package and paste the following inside it:

https://wiki.ros.org/ROS/Tutorials/WritingServiceClient%28c%2B%2B%29

More links:

https://wiki.ros.org/catkin/workspaces#Development_.28Devel.29_Space

```
#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;
}
```

Create the *src/add_two_ints_client.cpp* file within the beginner_tutorials package and paste the following inside it:

```
#include "ros/ros.h"
#include "beginner_tutorials/AddTwoInts.h"
#include <cstdlib>

int main(int argc, char **argv)
{
  ros::init(argc, argv, "add_two_ints_client");
  if (argc != 3)
  {
    ROS_INFO("usage: add_two_ints_client X Y");
    return 1;
  }
```

Codes continued next page.

https://wiki.ros.org/ROS/Tutorials/WritingServiceClient%28c%2B%2B%29

More links:

https://wiki.ros.org/catkin/workspaces#Development_.28Devel.29_Space

With codes previous page.

```
  ros::NodeHandle n;
  ros::ServiceClient client = n.serviceClient<beginner_tutorials::AddTwoInts>("add_two_ints");
  beginner_tutorials::AddTwoInts srv;
  srv.request.a = atoll(argv[1]);
  srv.request.b = atoll(argv[2]);
  if (client.call(srv))
  {
    ROS_INFO("Sum: %ld", (long int)srv.response.sum);
  }
  else
  {
    ROS_ERROR("Failed to call service add_two_ints");
    return 1;
  }

  return 0;
}
```

edit the beginner_tutorials CMakeLists.txt located at ~/catkin_ws/src/beginner_tutorials/CMakeLists.txt and add the following at the end:

```
add_executable(add_two_ints_server src/add_two_ints_server.cpp)
target_link_libraries(add_two_ints_server ${catkin_LIBRARIES})
add_dependencies(add_two_ints_server beginner_tutorials_gencpp)

add_executable(add_two_ints_client src/add_two_ints_client.cpp)
target_link_libraries(add_two_ints_client ${catkin_LIBRARIES})
add_dependencies(add_two_ints_client beginner_tutorials_gencpp)
```

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

```
$ roscore
```

```
$ rosrun beginner_tutorials add_two_ints_server
```

```
$ rosrun beginner_tutorials add_two_ints_client 1 3
```

Recording and playing back data

```
$ roscore
```

```
$ rosrun turtlesim turtlesim_node
```

```
$ rosrun turtlesim turtle_teleop_key
```

```
$ rostopic list -v
```

List the publishers and subscribers.

```
$ mkdir ~/bagfiles
```

```
$ cd ~/bagfiles
```

```
$ rosbag record -a
```

With the option -a, indicate that all published topics should be accumulated in a bag file..

https://wiki.ros.org/ROS/Tutorials/Recording%20and%20playing%20back%20data

More links:

https://wiki.ros.org/rosbag/Tutorials

$ rosbag info <your bagfile>	List info of your bag.
$ rosbag play <your bagfile>	Play back your bag.
$ rosbag play -r 2 <your bagfile>	Issue your keyboard commands twice as fast.
$ rosrun turtlesim turtlesim_node	
$ rosrun turtlesim turtle_teleop_key	
$ rosbag record -O subset /turtle1/cmd_vel /turtle1/pose	The -O records to log to *subset.bag*, and records to only subscribe to these two topics.
$ rosbag info subset.bag	List info of subset.bag.

More practises are essential.
Go now ~!