TECHIN 516: Robotic Sensing And Mobility

Lab 0: Linux introduction and ROS installation

DATE: Jan 5th. 2023

Lecturer: Queenie Qiu, John Raiti. Student: ***Shucheng Guo***

1 Linux Exercise

All the details of Linux you need are in the slides. Now we will use Linux for practice. Type the command you use for each problem. Each solution should be **one single** command.

1. Create a directory inside your home directory named Tech516Lab0.

   Solution:

   ```
   $ mkdir Tech516Lab0
   ```

2. Download our file lab0_linux.zip and save it into your new Tech516Lab0 directory by running:

   ```
   $ cd Tech516Lab0
   $ gdown --id 1kg1-A8zTnoSGl5ZKVx5ocvFi3z1A-xDz
   ```

   Unzip the '.zip file':

   ```
   $ unzip lab0_linux.zip
   ```

3. Assume you are in the *lab0_linux* directory, Copy the file *car_brain.py* from the current directory to the *src* subdirectory.

   Solution:

   ```
   $ cp car_brain.py src
   ```

4. List the files in the *lab0* directory, in "long listing format".

   Solution:

   ```
   $ ls -l
   ```

5. List all files, including hidden files, in the *src* directory, in reverse alphabetical order and long listing format.

   Solution:

   ```
   $ ls -lar lab0
   ```

6. Rename the file move_car.py to MoveCar.py
   hint: using mv.

   Solution:

   ```
   $ mv move_car.py MoveCar.py
   ```

7. Delete the files *tobedelete1.txt* and *tobedelete2.txt*

   Solution:

   ```
   $ rm tobedelete2.txt tobedelete1.txt
   ```

8. Self Discovery: You can use a * (asterisk) as a "wild-card" character to specify a group of files. For example, *foo means all files whose names end with foo, and foo* means all files whose names begin with foo. You can use a wildcard in the middle of a file name, such as foo*bar for all files that start with foo and end with bar.

   List all ".py" and ".txt" files in the current directory. Note that the ls command can accept more than one parameter for what files you want it to list.

   Solution:

   ```
   $ ls *.py *.txt
   ```

9. Self Discovery: What command can output the contents of a file to the terminal? Please use that command to output the contents of the file *CMakeLists.txt* and redirect the output to a new file called *redirection.txt* Write the single command below.

   Solution:

   ```
   $ cat CMakeLists.txt > redirection.txt
   ```

10. How many lines in the file *CMakeLists.txt* containing the word "cpp". Write the single command below.

    Solution:

    ```
    $ grep 'cpp' CMakeLists.txt | wc -l
    ```

2 ROS Tutorials

Please finish beginner level core ROS tutorials 1 to 17 on the ROS official website:
It is also where you can always look back for resources if you have any questions on ROS libraries and usages. Getting familiar with these ROS concepts will be very useful in your following studies.

2.1 Installing and Configuring Your ROS Environment

Please follow the instructions in tutorial 1 to set up your ROS environment and create a ROS workspace for catkin. Please take a screenshot for "echo $ROS_PACKAGE_PATH". **Your screen-shot is shown below**:

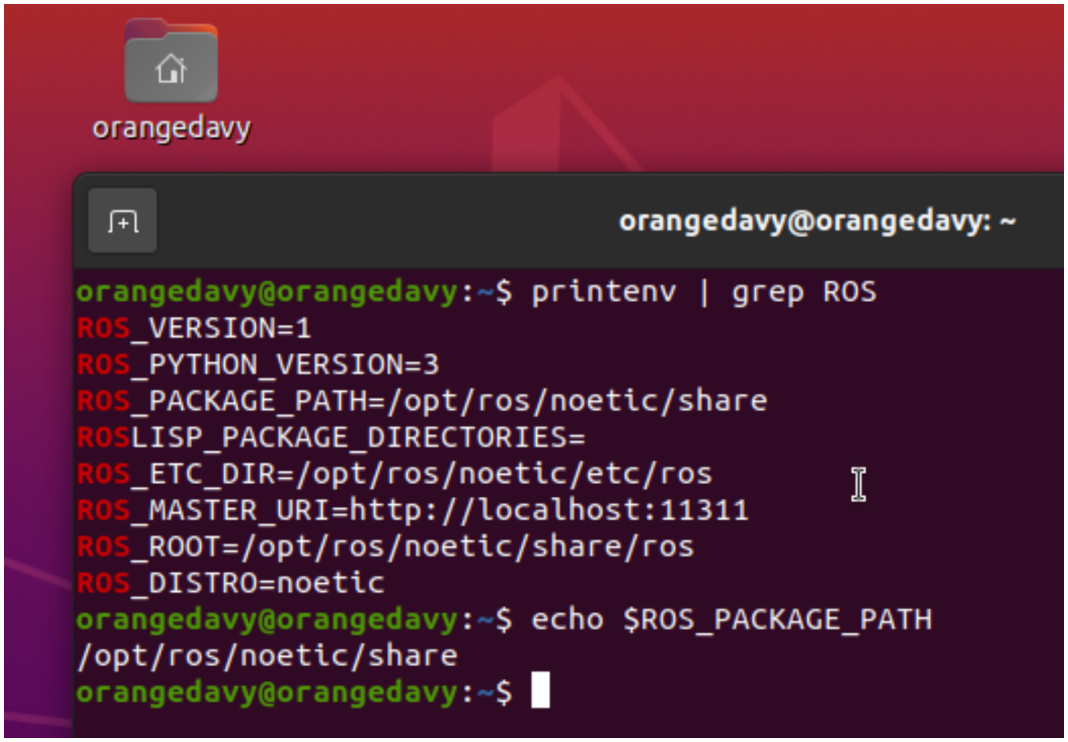

Figure 1: Printing `$ROS_PACKAGE_PATH`.

2.2 Navigating the ROS Filesystem

After soucing the setup.bash in your workspacex, please take a screen-shot of you using **rospack**, **roscd** and **rosls** and their results.Your screenshot is shown below:

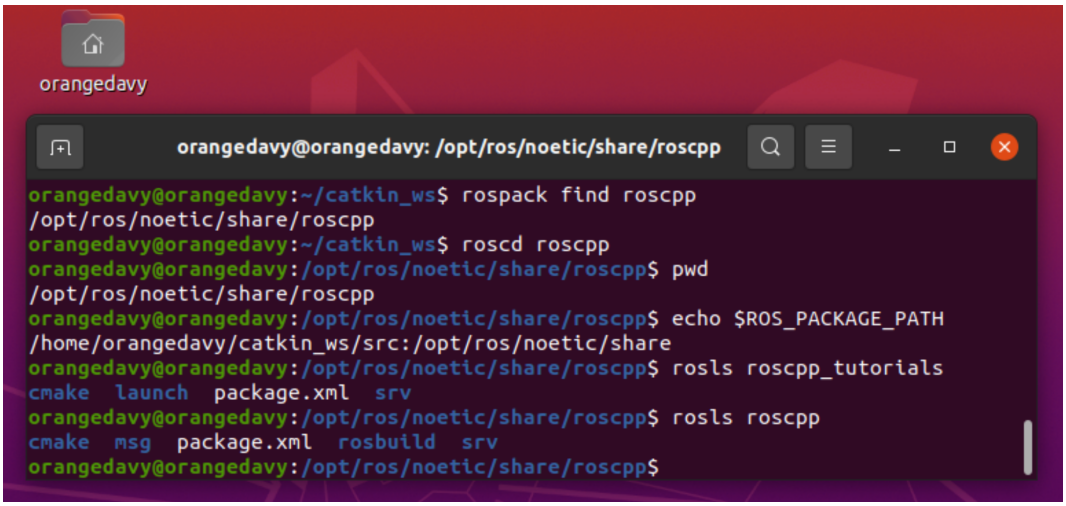

Figure 2: Using `rospack, roscd, rosls`.

2.3 Creating and Building a ROS package

Following tutorial 3 and 4, please provide a screenshot of the list of files in the src folder in the catkin workspace:

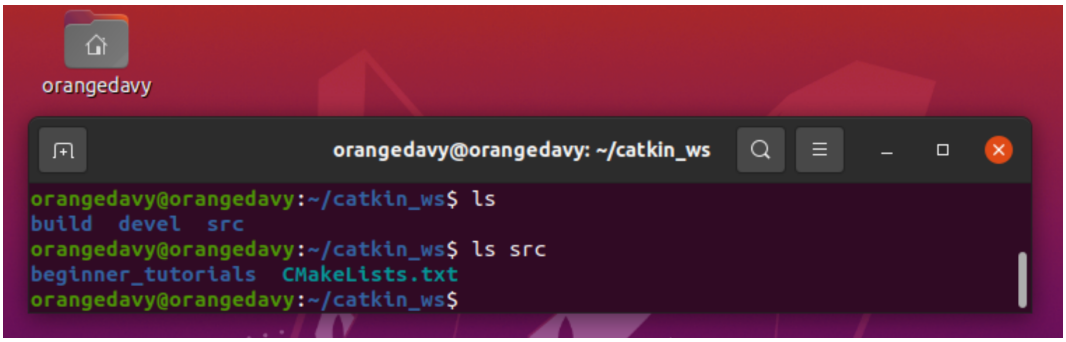

Figure 3: Checking `src` folder in catkin workspace.

2.4 Understanding ROS Nodes

Please answer the following questions:

1. How to run the ROS master.

   Solution:

   ```
   $ roscore
   ```

2. How to run the turtlesim_node.

   Solution:

   ```
   $ rosrun turtlesim turtlesim_node
   ```

3. How to check the list of running ROS nodes.

   Solution:

   ```
   $ rosnode list
   ```

2.5 Understanding ROS Topics

1. Use the node "turtlesim_node" and "turtle_teleop_key" to draw a path of your choice and take a screenshot. Your screenshot is shown below:

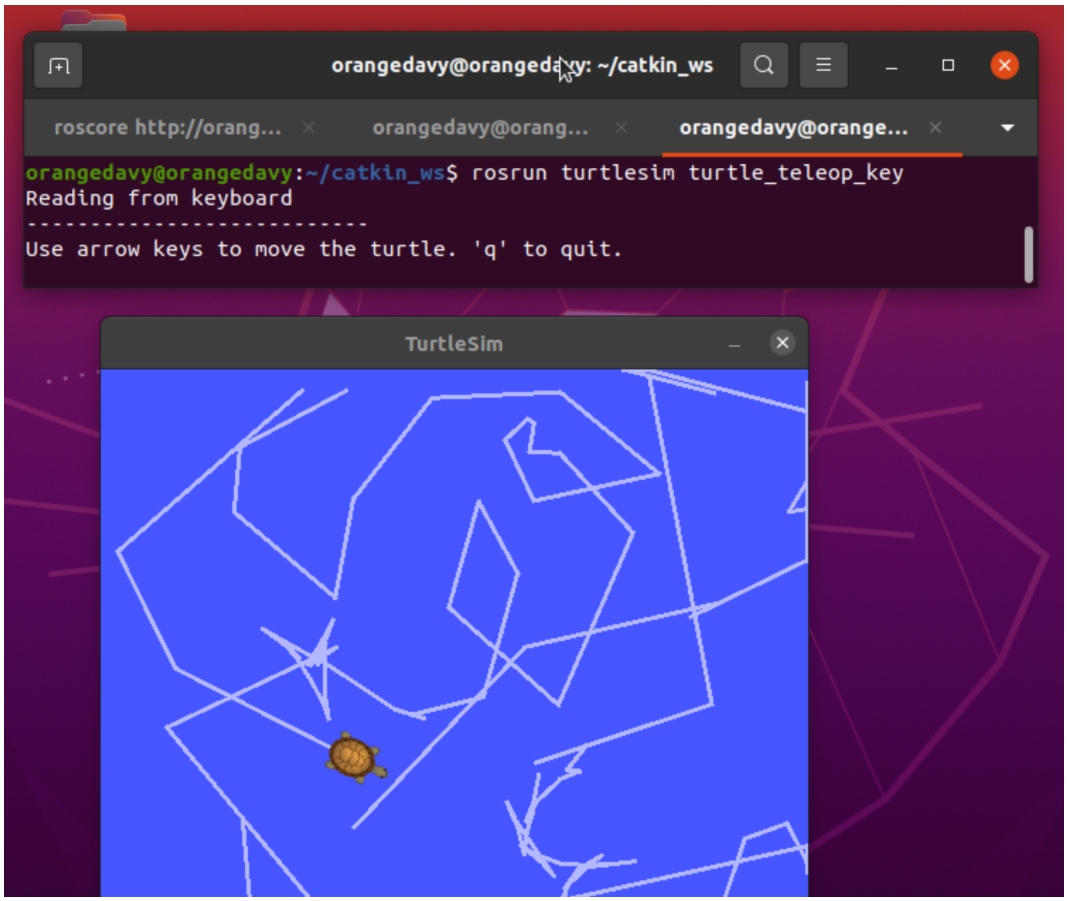

Figure 4: Drawing path with `turtlesim_node` and `turtle_teleop_key`.

2. Please briefly explain what is ROS topic, what is ROS message, their relationship and how to check the type of topic.

   (a) **ROS topic**: ROS topic transports information between nodes. The information is organized as a data structure and can have different data types. Topics can be identified by their name and their type.

(b) **ROS message**: ROS message is the information sent by the topic between nodes. To communicate, the publisher and subscriber need the same type of message, which defines the type of its topic.

(c) **Check type**: `$ rostopic type [topic]`

2.6 Understanding ROS Services and Parameters

1. Use the spawn service in turtlesim_node to create a new turtle at **(x,y) = 2, 3; theta = 0.5; name = "turtleNo2"** and take a screenshot of TurtleSim window:

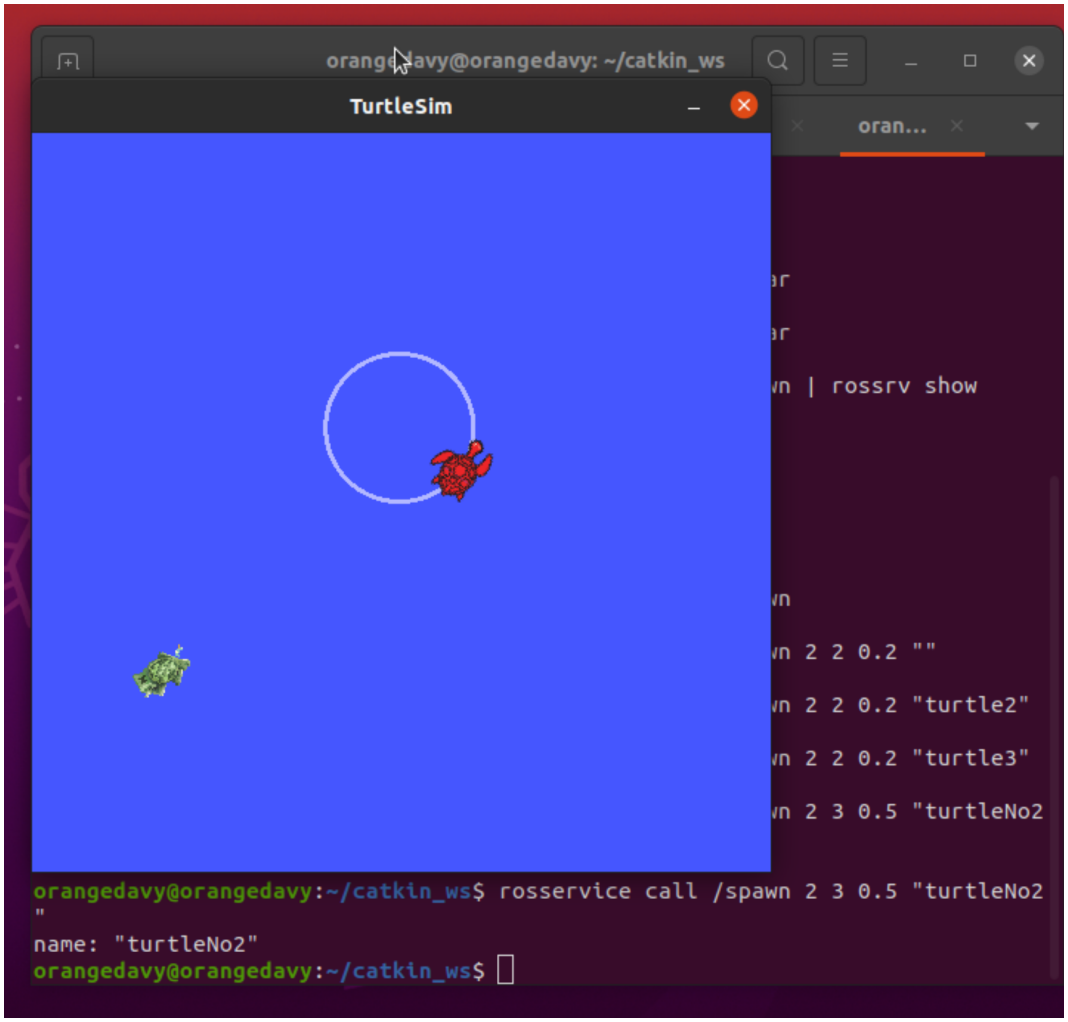

Figure 5: Spawning the new `turtleNo2` instance.

2. Figure out how to use the /kill service and use it to remove the turtle you created in the question 1. Please write down the command:

```
$ rosservice call /kill "name:'turtleNo2'"
```

2.7 Using roslaunch

There is an example launch file in the Lab0.zip. Check the content and roslaunch it. Please explain what does the launch file do? What are the benefits of using launch files/roslaunch?

Command: `$ roslaunch example.launch`

Explanation: The example launch file creates a new turtle named `turtlesim_node` with a node in the package `turtlesim`, while enables another that allows the user to move the turtle around with arrow keys under the name `turtle_teleop_key`.

Benefits: Using launch files is a tidy way for the user to initiate several nodes with desired configurations and parameters all at once.

2.8 Creating a ROS msg and srv

As we have learned ROS messages and ROS services, here we will learn how to write customized messages and services.

1. Please follow the tutorial to create a msg file named Num in the beginner_tutorials package. Modify the package.xml and CMakeLists.txt, and provide a screenshot of the result of

   ```
   $ rosmsg show Num
   ```

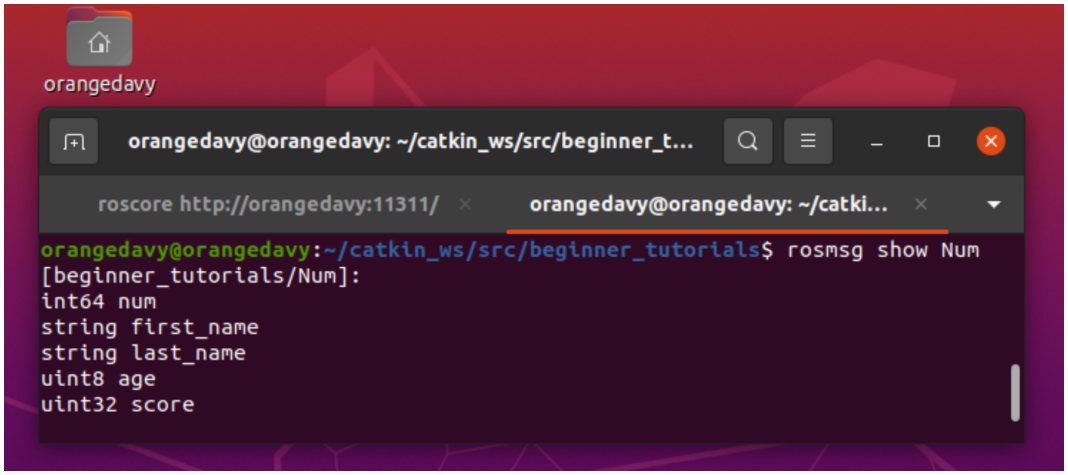

Figure 6: Creating a `msg` file `Num`.

2. Follow the tutorial to create a srv named AddTwoInts in beginner_tutorials. Then provide a screenshot of the result of

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

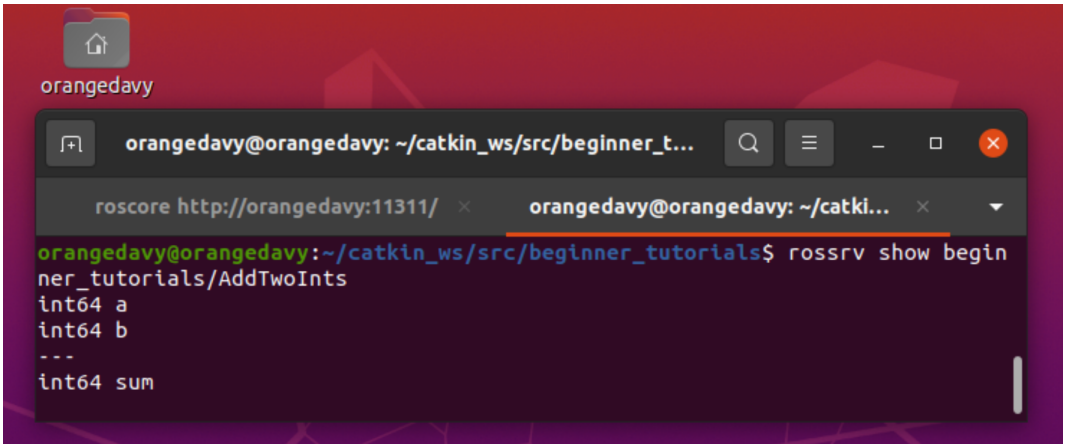

Figure 7: Creating a `srv` file `AddTwoInts`.

2.9 Writing a Simple Publisher and Subscriber (Python)

Please briefly explain what does the sample codes do? What are the names of the publisher and subscriber nodes? What's name of the topic? What's the topic type?

Explanation: The sample codes are divided into a Publisher and a Subscriber. The former creates a publisher nodes that publishes to a certain topic a certain type of message at a desired rate until it stops. The latter utilizes the callback mechanism to subscribe to the messages published to the topic as it works.

Nodes: Publisher name: `talker`, Subscriber name: `listener`.

Topic: Topic is named `chatter`, of the type `std_msgs.msgs.String`.

2.10 Examining the Simple Publisher and Subscriber

After running the two nodes, let's do one more step. Open a new window, echo the topic that is published/subscribed by the nodes and take a screenshot:

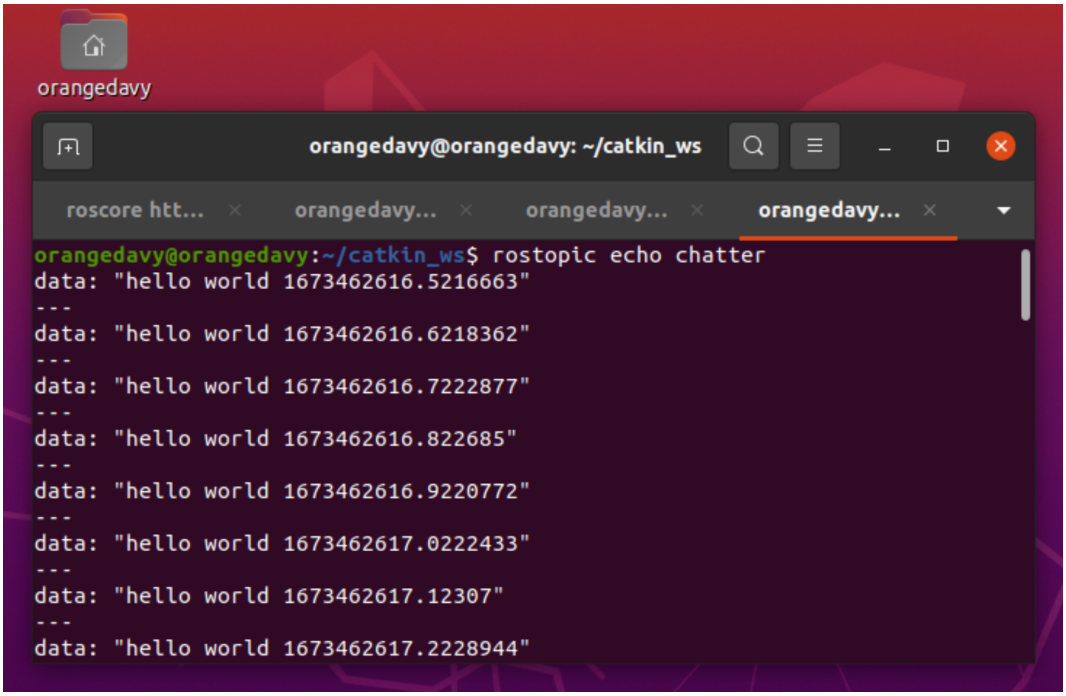

Figure 8: Echoing the messages published to `chatter`.

2.11 Writing a Simple Service and Client (Python)

What's name of the service? What's the service type?

Service: Service name is `add_two_ints`, of the type `AddTwoInts`.

2.12 Examining the Simple Service and Client

After finishing the tutorial, you will have an idea about how the server and client works. When you write your own service, you may want to examine that your service runs properly before writing the client. As the add_two_ints_server is running, try to examine that service runs as expected using 2 plus 3 as a test case and take a screenshot. (Hint: use rosservice)

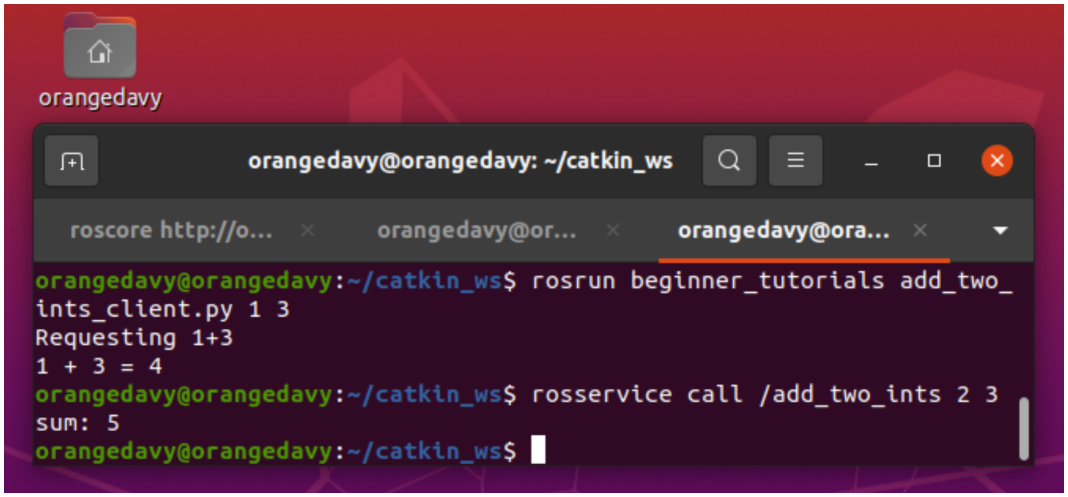

Figure 9: Testing the service with `2 + 3`.

2.13 Recording and playing back data

Follow the tutorial 17, record data of all topics for around 10 seconds. After this step, please take a screenshot of the info of your recorded bag file:

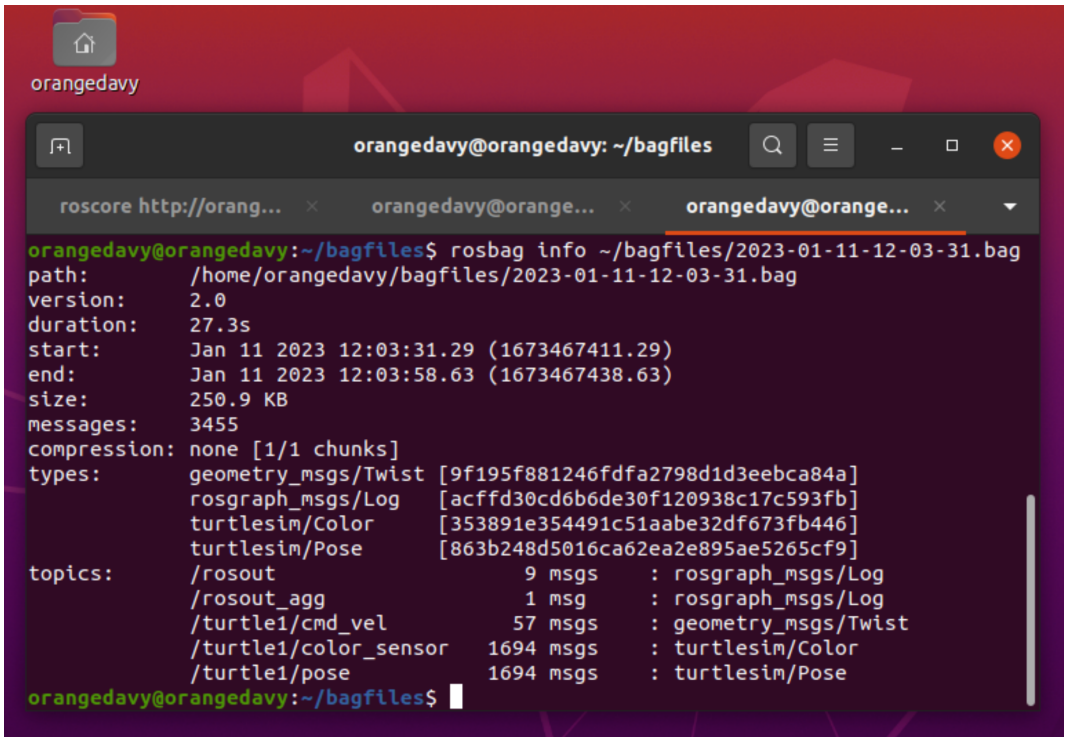

Figure 10: Recording bag files with `rosbag info`.