ECE 478-578
Intelligent Robotics I

PhD. Husnu Melih Erdogan – Electrical & Computer Engineering

herdogan@pdx.edu Teaching Assistant

Portland State
UNIVERSITY

Introduction to ROS Part - 3

ROS
(Robot Operating System)

Portland State
UNIVERSITY

Course Structure

- **Part 1 - Overview**
  - What is ROS?
  - Introduction to ROS
  - ROS architecture, philosophy, history
  - How to install ROS?
  - Examples
  - Installation
  - ROS Master
  - ROS Nodes
  - ROS Topic
  - ROS Messages
  - Console Commands
  - ROS Packages
  - ROS Launch-files
  - Catkin Workspace and Build System
  - Turtlesim

- **Part 2 - Details**
  - ROS File System
  - ROS Package
  - How to create a package?
  - How to build a package?
  - Creating a Publisher Node
  - Creating a Subscriber Node

- **Part 3 - Details**
  - Publisher and Subscriber Node
  - Creating a Launch File
  - How to use ROS .bagfiles?
  - rqt_bag
  - ROS Parameters
  - Assignment 3
  - Assignment 4

- **Part 4 – Mini Project**
  - Rviz
  - ROS Services
  - ROS Actions
  - ROS Massages
  - Fuzzy Logic
  - 2D Multi-Robot Simulator
  - Assignment 5

Portland State UNIVERSITY

ROS (Robot Operating System Review)

Creating a Subscriber Node

Portland State
UNIVERSITY

Creating a Subscriber Node

- Go to your package in workspace
  - **cd catkin_ws/src/mypackage**
- Create directory called “scripts”
  - **mkdir scripts**
- Go in to scripts
  - **cd scripts**
- Create script with your favirote editor “listener.py”
- Make the python script executable
  - **chmod +x listener.py**

Portland State UNIVERSITY

Creating a Subscriber Node

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    rospy.spin()

if __name__ == '__main__':
    listener()
```

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # node are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

It tells your system this a Python file
It allows us to use Python with ROS
Import message type string, so we can reuse it

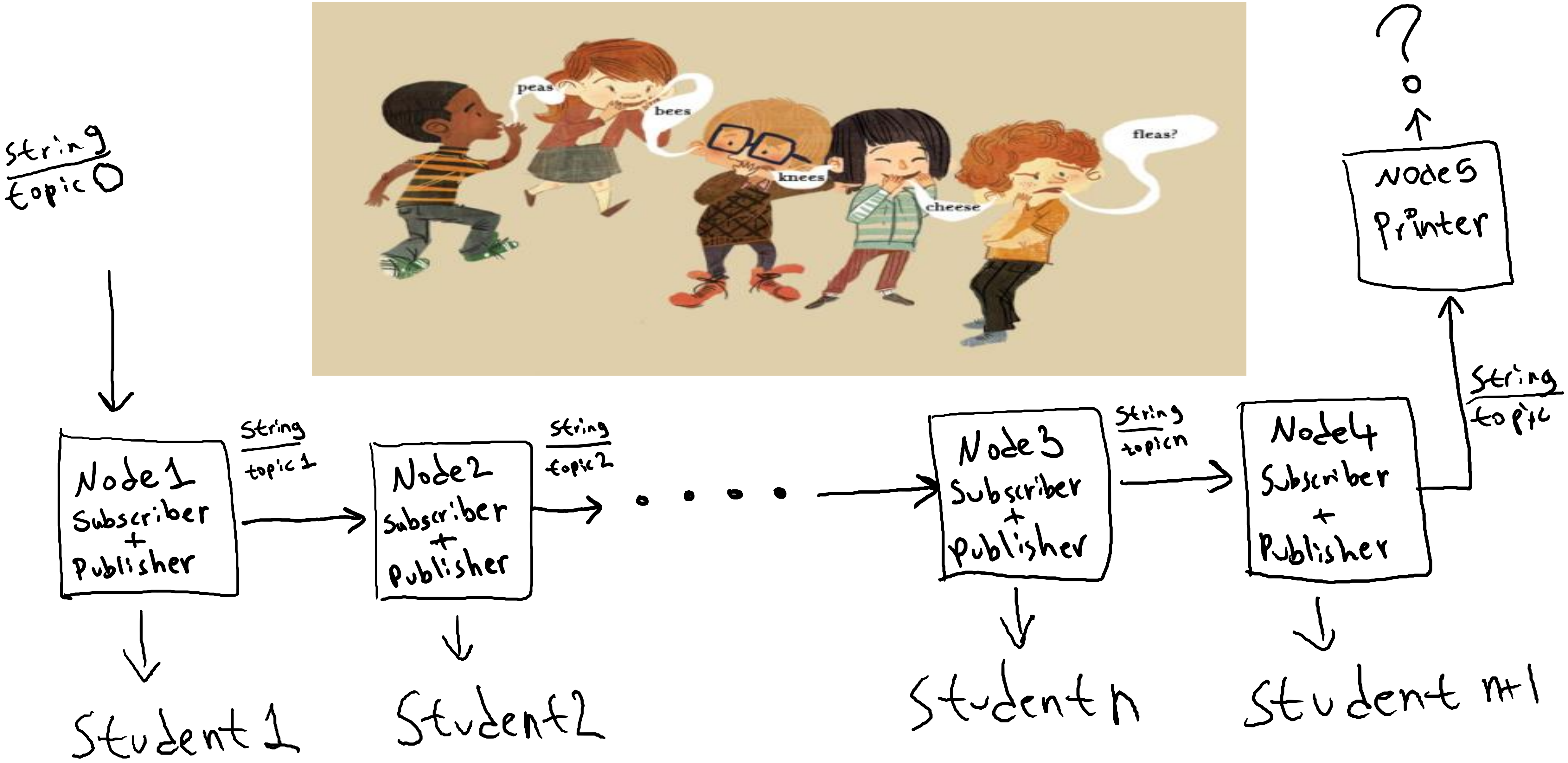
It prints to screen as it also writes to stdout

Create a node called listener

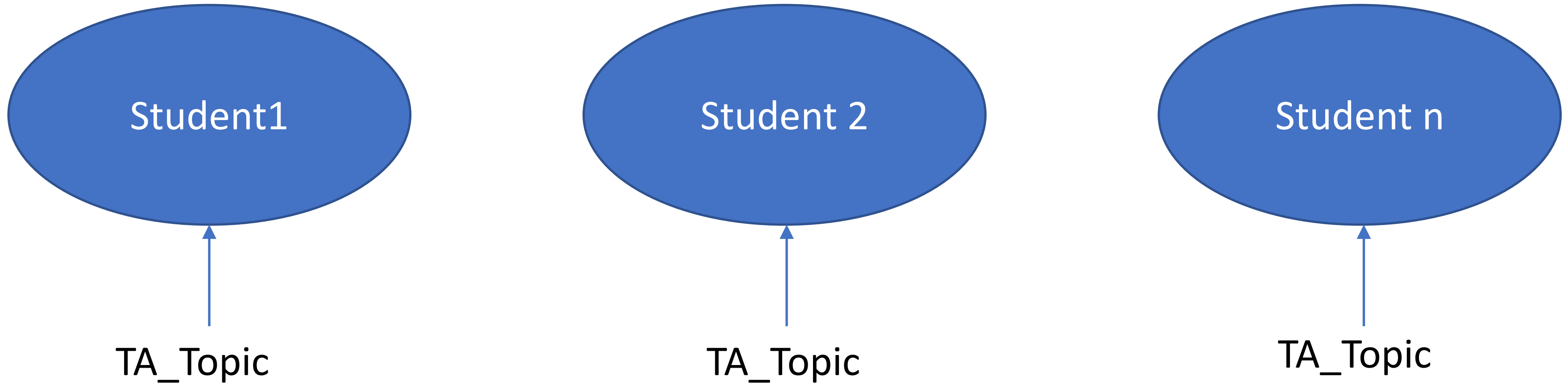
Node subscribes to the chatter.
When new messages are received
callback method is invoked.

1 2 3

Example 1 Telephone Game

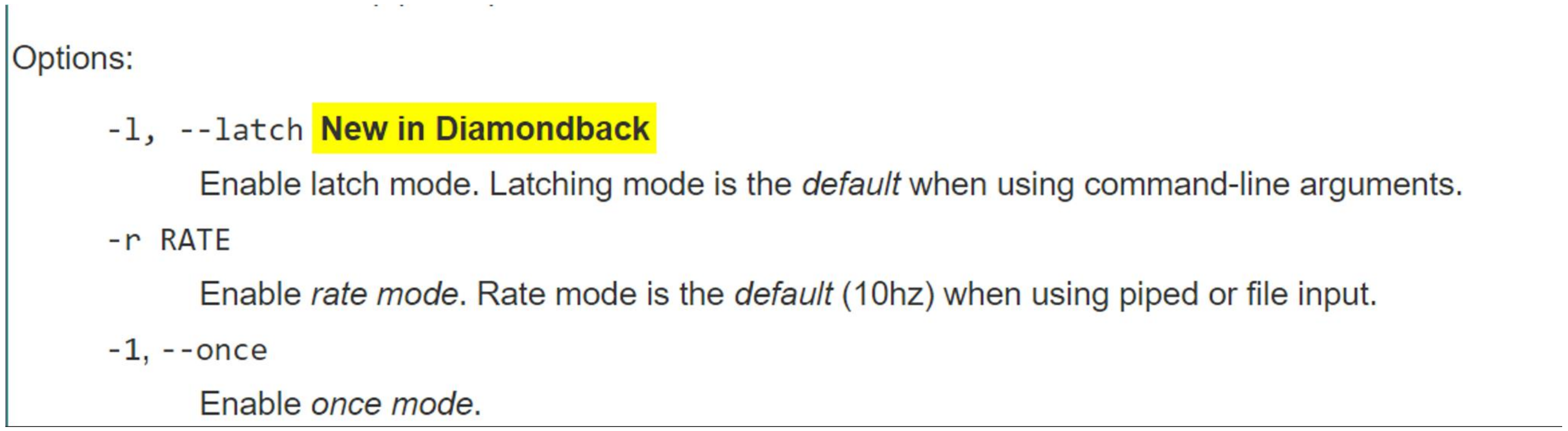


Telephone – Step1



Let's Test Our Node

- Publishes data to a topic
- rostopic pub /topic_name std_msgs/String Perkowski



Options:

```
    -l, --latch
```
New in Diamondback

Enable latch mode. Latching mode is the *default* when using command-line arguments.

```
    -r RATE
```
Enable *rate mode*. Rate mode is the *default* (10hz) when using piped or file input.

```
    -1, --once
```
Enable *once mode*.

- Publishes data to a topic
- rostopic pub /TA_topic std_msgs/String Hello World

Creating a Publisher Node

Portland State
UNIVERSITY

Creating a Publisher Node

- Go to your package in workspace
  - **cd catkin_ws/src/mypackage**
- Create directory called “scripts”
  - **mkdir scripts**
- Go in to scripts
  - **cd scripts**
- Create script with your favorite editor “talker.py”
- Make the python script executable
  - **chmod +x talker.py**

Portland State UNIVERSITY

Publisher Node

```python
#!/usr/bin/env python
# license removed for brevity
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Publisher Node

```python
#!/usr/bin/env python
# license removed for brevity
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Topic Name - Message Type - If messages are not received

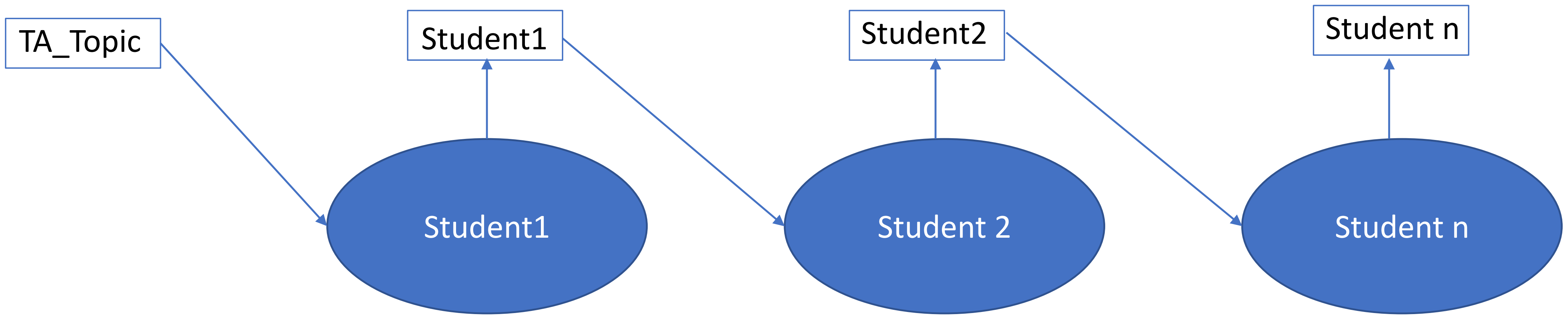
10 times per second

Node name: talker / anonymous = True adds random number to your node name

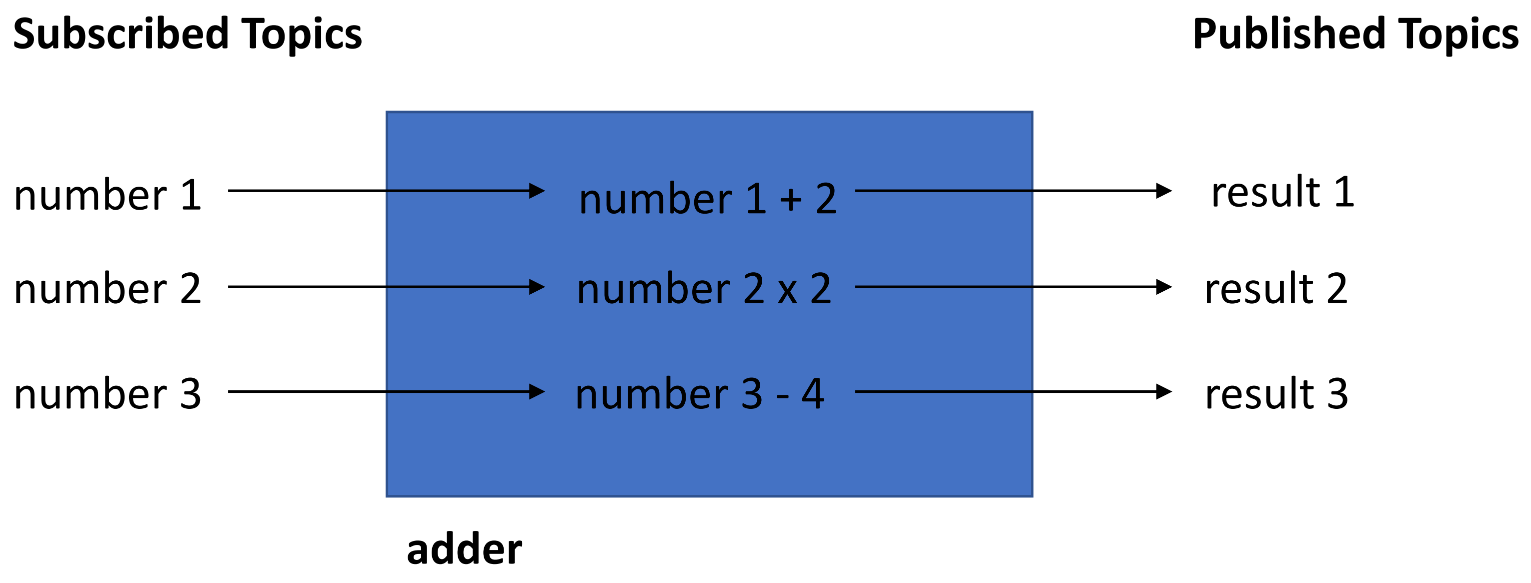
Publish the topic

In order to make sure that it doesn't keep executing the code, use exeption

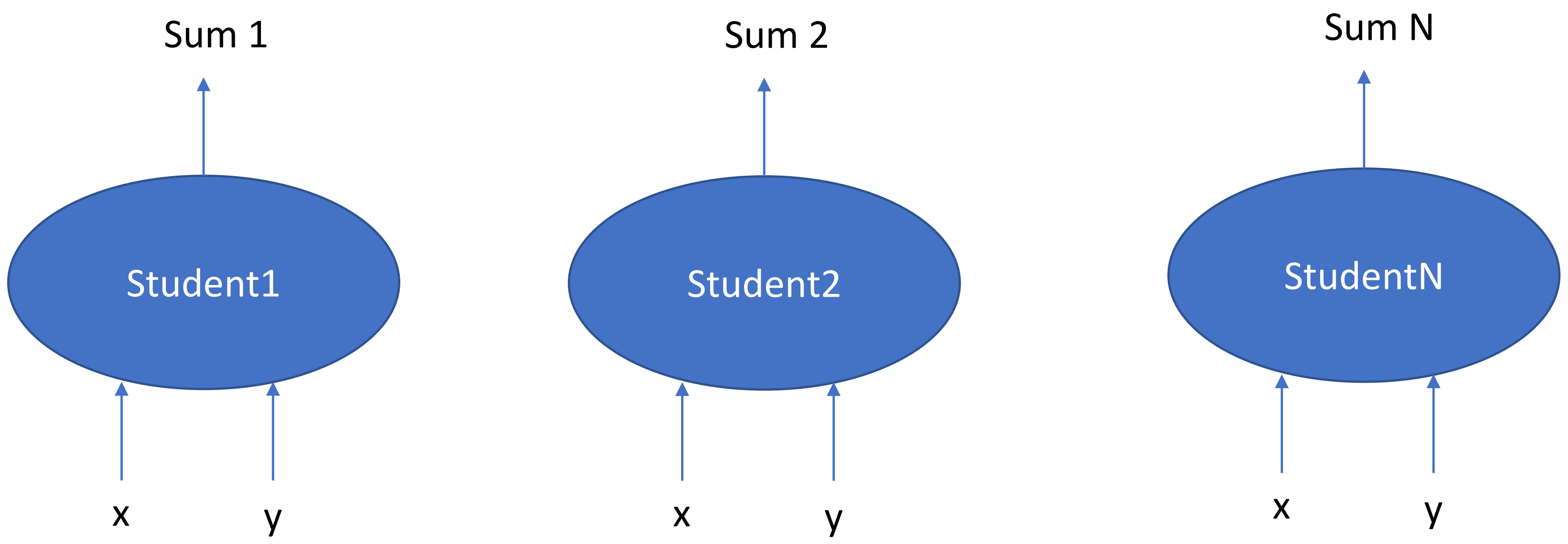
Telephone – Step 2



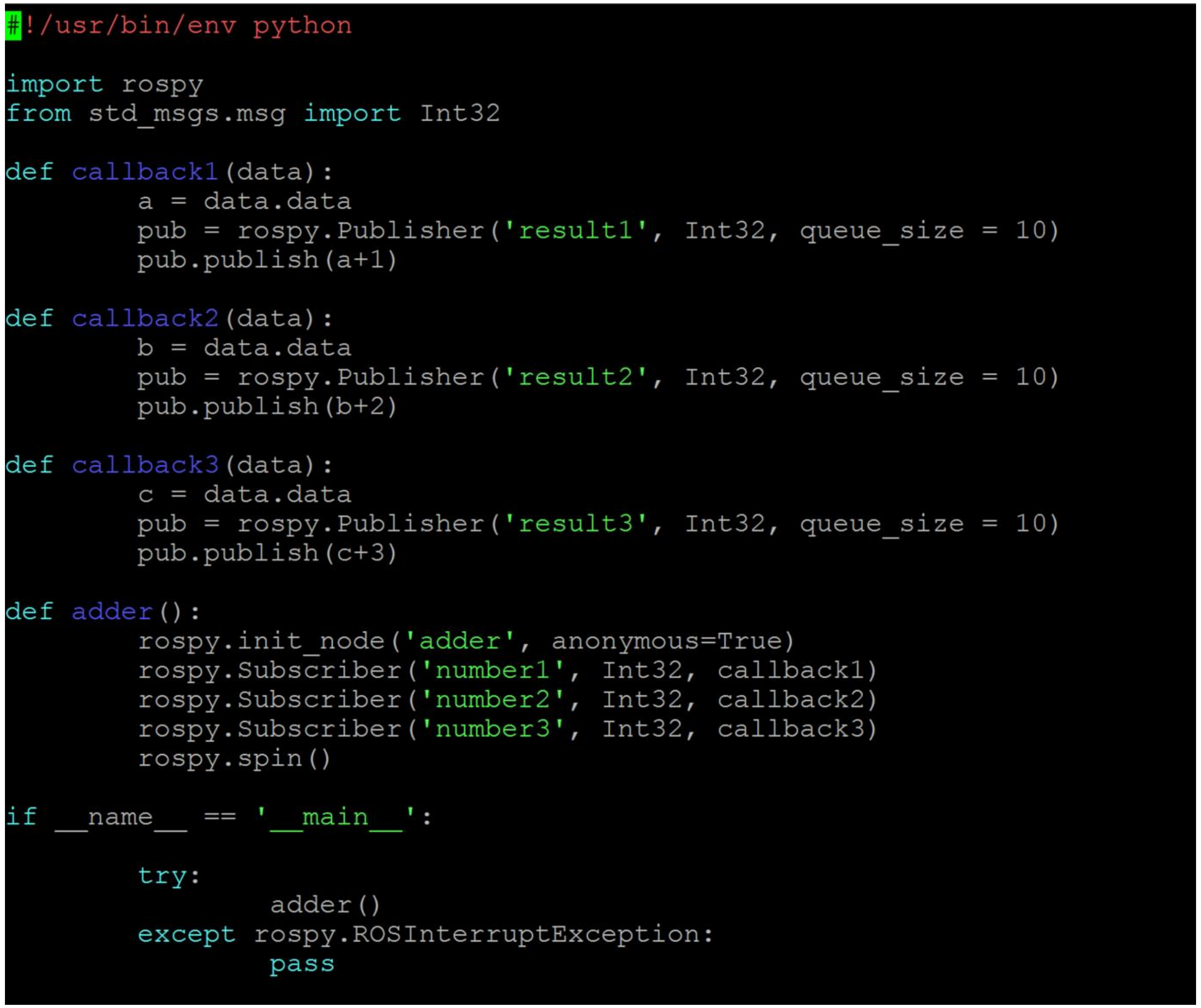
Example 2 Subscribe and Publish Multiple Topics



Tutorial 2 - Calculator



Tutorial 2 - Calculator



```
#!/usr/bin/env python

import rospy
from std_msgs.msg import Int32

def callback1(data):
        a = data.data
        pub = rospy.Publisher('result1', Int32, queue_size = 10)
        pub.publish(a+1)

def callback2(data):
        b = data.data
        pub = rospy.Publisher('result2', Int32, queue_size = 10)
        pub.publish(b+2)

def callback3(data):
        c = data.data
        pub = rospy.Publisher('result3', Int32, queue_size = 10)
        pub.publish(c+3)

def adder():
        rospy.init_node('adder', anonymous=True)
        rospy.Subscriber('number1', Int32, callback1)
        rospy.Subscriber('number2', Int32, callback2)
        rospy.Subscriber('number3', Int32, callback3)
        rospy.spin()

if __name__ == '__main__':

        try:
                adder()
        except rospy.ROSInterruptException:
                pass
```

Portland State
UNIVERSITY

ROS Launch File

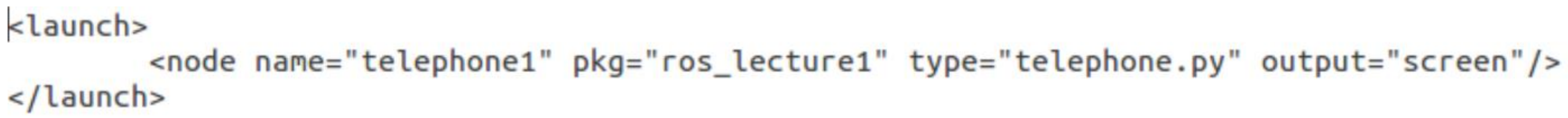
Portland State UNIVERSITY

ROS Launch

- roslaunch is a tool for easily launching multiple ROS nodes locally and remotely
- It also allows you to set parameters on the Parameter Server.

- How to run a launch file:
- roslaunch package_name file.launch

Simple Launch File



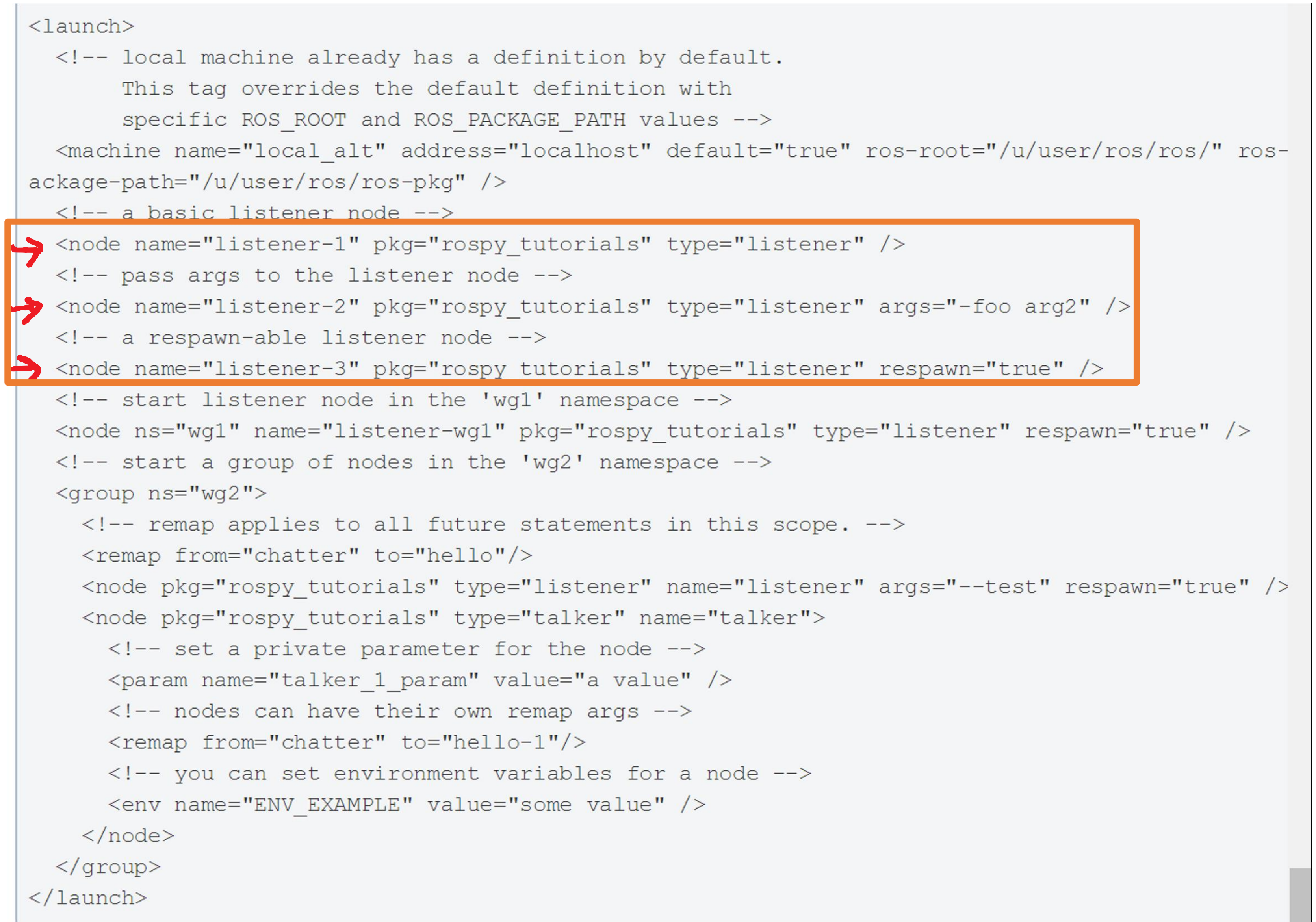
```
<launch>
        <node name="telephone1" pkg="ros_lecture1" type="telephone.py" output="screen"/>
</launch>
```

Example Launch File

```
<launch>
  <node
    pkg="turtlesim"
    type="turtlesim_node"
    name="turtlesim"
    respawn="true"
  />
  <node
    pkg="turtlesim"
    type="turtle_teleop_key"
    name="teleop_key"
    required="true"
    launch-prefix="xterm -e"
  />
  <node
    pkg="agitr"
    type="subpose"
    name="pose_subscriber"
    output="screen"
  />
</launch>
```

Listing 6.1: A launch file called example.launch that starts three nodes at once.

A Little bit more complicated one



```
<launch>
  <!-- local machine already has a definition by default.
       This tag overrides the default definition with
       specific ROS_ROOT and ROS_PACKAGE_PATH values -->
  <machine name="local_alt" address="localhost" default="true" ros-root="/u/user/ros/ros/" ros-
ackage-path="/u/user/ros/ros-pkg" />
  <!-- a basic listener node -->
  <node name="listener-1" pkg="rospy_tutorials" type="listener" />
  <!-- pass args to the listener node -->
  <node name="listener-2" pkg="rospy_tutorials" type="listener" args="-foo arg2" />
  <!-- a respawn-able listener node -->
  <node name="listener-3" pkg="rospy_tutorials" type="listener" respawn="true" />
  <!-- start listener node in the 'wg1' namespace -->
  <node ns="wg1" name="listener-wg1" pkg="rospy_tutorials" type="listener" respawn="true" />
  <!-- start a group of nodes in the 'wg2' namespace -->
  <group ns="wg2">
    <!-- remap applies to all future statements in this scope. -->
    <remap from="chatter" to="hello"/>
    <node pkg="rospy_tutorials" type="listener" name="listener" args="--test" respawn="true" />
    <node pkg="rospy_tutorials" type="talker" name="talker">
      <!-- set a private parameter for the node -->
      <param name="talker_1_param" value="a value" />
      <!-- nodes can have their own remap args -->
      <remap from="chatter" to="hello-1"/>
      <!-- you can set environment variables for a node -->
      <env name="ENV_EXAMPLE" value="some value" />
    </node>
  </group>
</launch>
```

- **respawn**="true"*(optional)* :Restart the node automatically if it quits.
- **respawn_delay**="30" *(optional, default 0)* **New in ROS indigo :** If respawn is true, wait respawn_delay seconds after the node failure is detected before attempting restart.

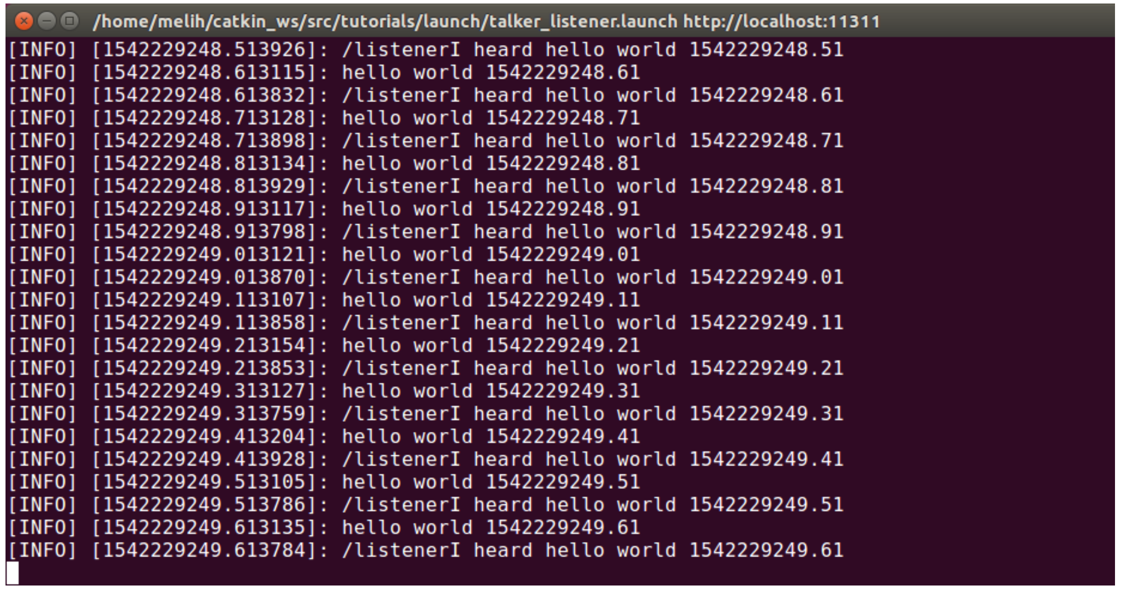
Launch file xml tag references

- <launch>
- <node>
- <machine>
- <include>
- <remap>
- <env>
- <param>
- <rosparam>
- <group>
- <test>
- <arg>

ROS Launch File Example

- Rosluanch tutorials talker_listener.launch

```
<launch>
  <node name="listener" pkg="rospy_tutorials" type="listener.py" output="screen"/>
  <node name="talker" pkg="rospy_tutorials" type="talker.py" output="screen"/>
</launch>
```



/home/melih/catkin_ws/src/tutorials/launch/talker_listener.launch http://localhost:11311

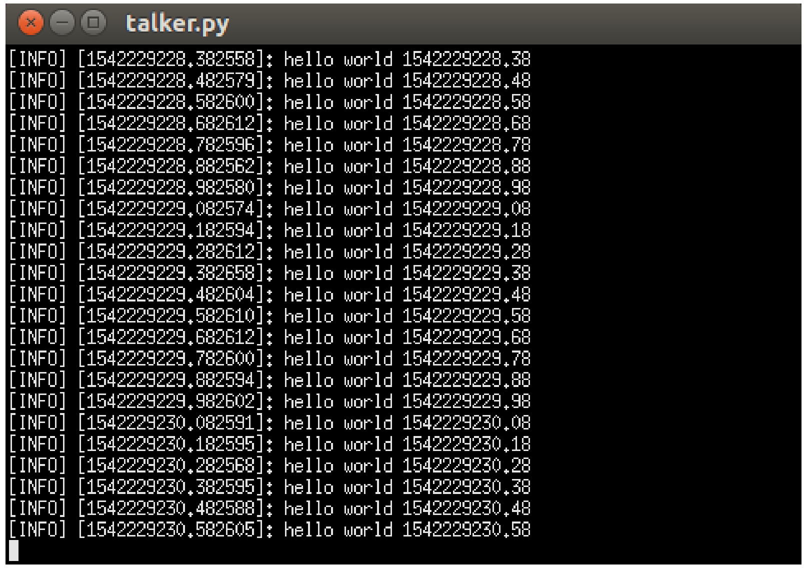
```
[INFO] [1542229248.513926]: /listenerI heard hello world 1542229248.51
[INFO] [1542229248.613115]: hello world 1542229248.61
[INFO] [1542229248.613832]: /listenerI heard hello world 1542229248.61
[INFO] [1542229248.713128]: hello world 1542229248.71
[INFO] [1542229248.713898]: /listenerI heard hello world 1542229248.71
[INFO] [1542229248.813134]: hello world 1542229248.81
[INFO] [1542229248.813929]: /listenerI heard hello world 1542229248.81
[INFO] [1542229248.913117]: hello world 1542229248.91
[INFO] [1542229248.913798]: /listenerI heard hello world 1542229248.91
[INFO] [1542229249.013121]: hello world 1542229249.01
[INFO] [1542229249.013870]: /listenerI heard hello world 1542229249.01
[INFO] [1542229249.113107]: hello world 1542229249.11
[INFO] [1542229249.113858]: /listenerI heard hello world 1542229249.11
[INFO] [1542229249.213154]: hello world 1542229249.21
[INFO] [1542229249.213853]: /listenerI heard hello world 1542229249.21
[INFO] [1542229249.313127]: hello world 1542229249.31
[INFO] [1542229249.313759]: /listenerI heard hello world 1542229249.31
[INFO] [1542229249.413204]: hello world 1542229249.41
[INFO] [1542229249.413928]: /listenerI heard hello world 1542229249.41
[INFO] [1542229249.513105]: hello world 1542229249.51
[INFO] [1542229249.513786]: /listenerI heard hello world 1542229249.51
[INFO] [1542229249.613135]: hello world 1542229249.61
[INFO] [1542229249.613784]: /listenerI heard hello world 1542229249.61
```

Portland State UNIVERSITY

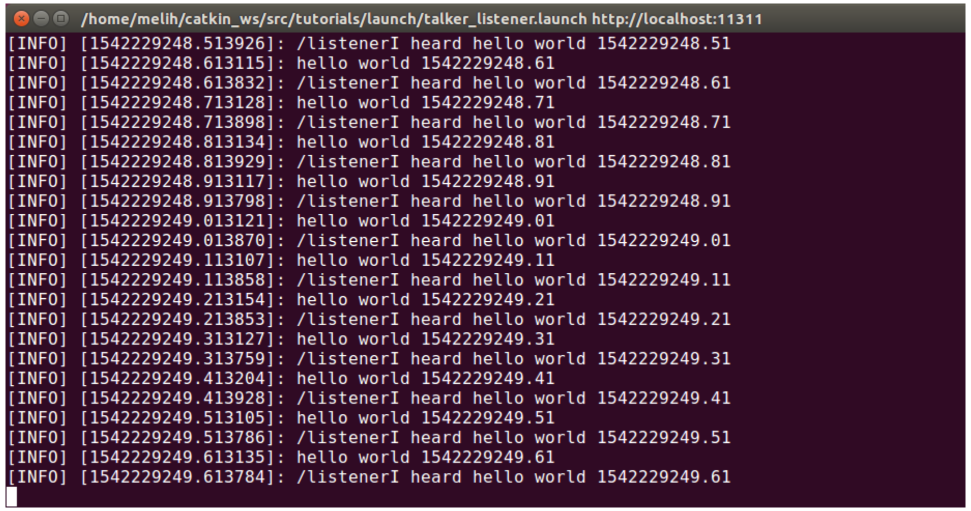
ROS Launch File Example

- You can also launch a node in a new terminal
  - launch-prefix=“xterm –e”

```
<launch>
  <node name="listener" pkg="rospy_tutorials" type="listener.py" output="screen"/>
  <node name="talker" pkg="rospy_tutorials" type="talker.py" output="screen"/>
  <node ns="t1" name="talker" pkg="rospy_tutorials" type="talker.py" launch-prefix="xterm -e"/>
</launch>
```


talker.py

```
[INFO] [1542229228.382558]: hello world 1542229228.38
[INFO] [1542229228.482579]: hello world 1542229228.48
[INFO] [1542229228.582600]: hello world 1542229228.58
[INFO] [1542229228.682612]: hello world 1542229228.68
[INFO] [1542229228.782596]: hello world 1542229228.78
[INFO] [1542229228.882562]: hello world 1542229228.88
[INFO] [1542229228.982580]: hello world 1542229228.98
[INFO] [1542229229.082574]: hello world 1542229229.08
[INFO] [1542229229.182594]: hello world 1542229229.18
[INFO] [1542229229.282612]: hello world 1542229229.28
[INFO] [1542229229.382658]: hello world 1542229229.38
[INFO] [1542229229.482604]: hello world 1542229229.48
[INFO] [1542229229.582610]: hello world 1542229229.58
[INFO] [1542229229.682612]: hello world 1542229229.68
[INFO] [1542229229.782600]: hello world 1542229229.78
[INFO] [1542229229.882594]: hello world 1542229229.88
[INFO] [1542229229.982602]: hello world 1542229229.98
[INFO] [1542229230.082591]: hello world 1542229230.08
[INFO] [1542229230.182595]: hello world 1542229230.18
[INFO] [1542229230.282568]: hello world 1542229230.28
[INFO] [1542229230.382595]: hello world 1542229230.38
[INFO] [1542229230.482588]: hello world 1542229230.48
[INFO] [1542229230.582605]: hello world 1542229230.58
```


/home/melih/catkin_ws/src/tutorials/launch/talker_listener.launch http://localhost:11311

```
[INFO] [1542229248.513926]: /listenerI heard hello world 1542229248.51
[INFO] [1542229248.613115]: hello world 1542229248.61
[INFO] [1542229248.613832]: /listenerI heard hello world 1542229248.61
[INFO] [1542229248.713128]: hello world 1542229248.71
[INFO] [1542229248.713898]: /listenerI heard hello world 1542229248.71
[INFO] [1542229248.813134]: hello world 1542229248.81
[INFO] [1542229248.813929]: /listenerI heard hello world 1542229248.81
[INFO] [1542229248.913117]: hello world 1542229248.91
[INFO] [1542229248.913798]: /listenerI heard hello world 1542229248.91
[INFO] [1542229249.013121]: hello world 1542229249.01
[INFO] [1542229249.013870]: /listenerI heard hello world 1542229249.01
[INFO] [1542229249.113107]: hello world 1542229249.11
[INFO] [1542229249.113858]: /listenerI heard hello world 1542229249.11
[INFO] [1542229249.213154]: hello world 1542229249.21
[INFO] [1542229249.213853]: /listenerI heard hello world 1542229249.21
[INFO] [1542229249.313127]: hello world 1542229249.31
[INFO] [1542229249.313759]: /listenerI heard hello world 1542229249.31
[INFO] [1542229249.413204]: hello world 1542229249.41
[INFO] [1542229249.413928]: /listenerI heard hello world 1542229249.41
[INFO] [1542229249.513105]: hello world 1542229249.51
[INFO] [1542229249.513786]: /listenerI heard hello world 1542229249.51
[INFO] [1542229249.613135]: hello world 1542229249.61
[INFO] [1542229249.613784]: /listenerI heard hello world 1542229249.61
```

Parameter YAML Format and Examples

- We can also store dictionaries on the parameter server.
- If the number of parameters is high, we can use a YAML file to save it.
- YAML™ is a human-friendly, cross language, Unicode based data serialization language designed around the common native data structures of agile programming languages.
- It is broadly useful for programming needs ranging from configuration files to Internet messaging to object persistence to data auditing.
- Here is an example of the YAML file parameter definitions:

```yaml
string: 'foo'
integer: 1234
float: 1234.5
boolean: true
list: [1.0, mixed list]
dictionary: {a: b, c: d}
```

```yaml
/camera/name : 'nikon'     #string type
/camera/fps : 30           #integer
/camera/exposure : 1.2     #float
/camera/active : true      #boolean
```

Portland State UNIVERSITY

Parameter Server

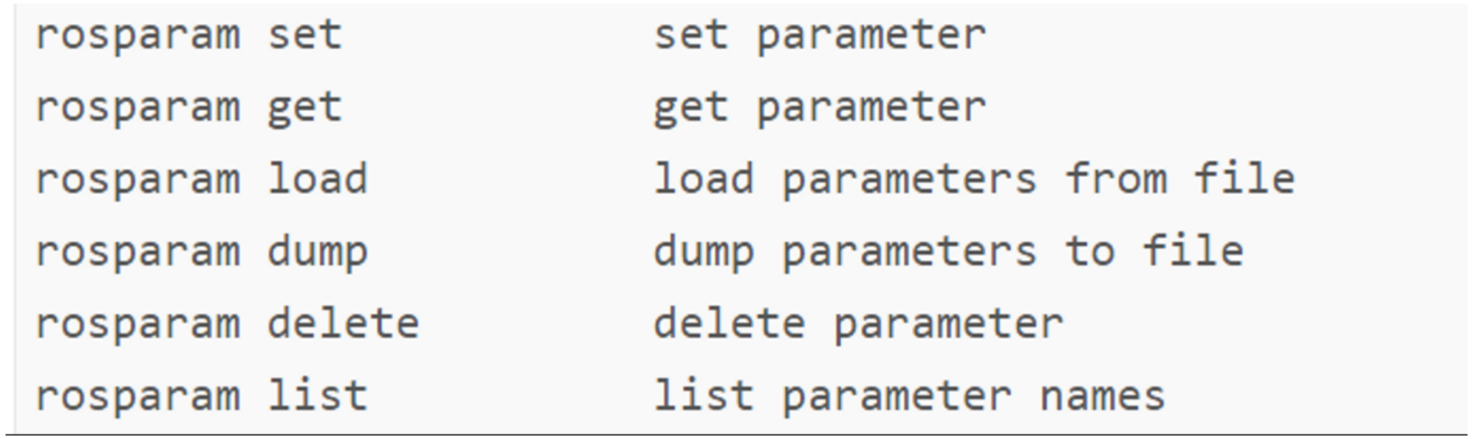
- In addition to the messages ROS provides another mechanism called parameters to get information to nodes.
- Configuration information in ROS is usually saved to the Parameter server. The Parameter sever is a collection of values that can be accessed upon request through the command prompt, nodes or launch files.
- Parameters are intended to be fairly static, globally available values such as integers, floats, strings or bool values.
- Any node can set parameters
- Any node can have access to parameters

- **rospy.set_param(param_name, param_value)**

Portland State UNIVERSITY

Parameter Server Usage

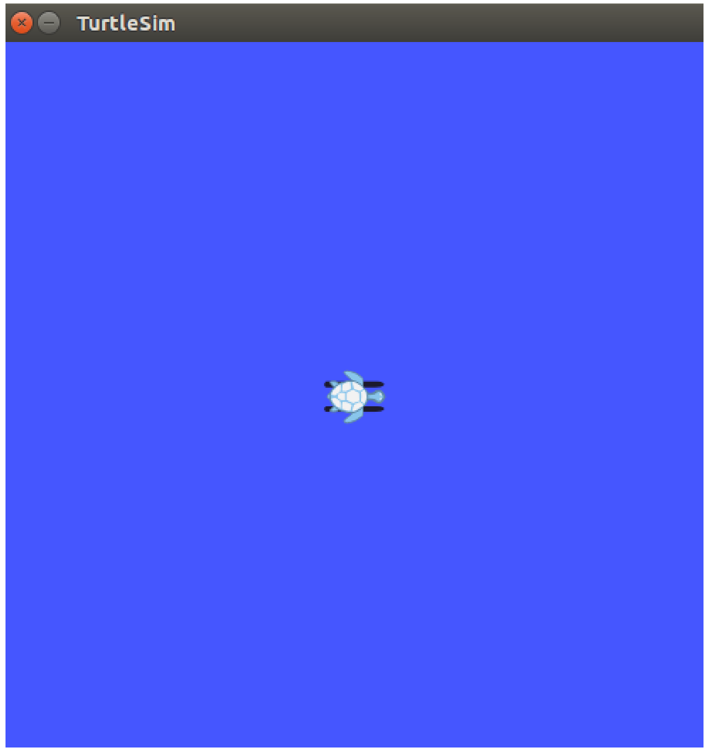
- When you run roscore to start ROS Master, it also starts parameter server too.

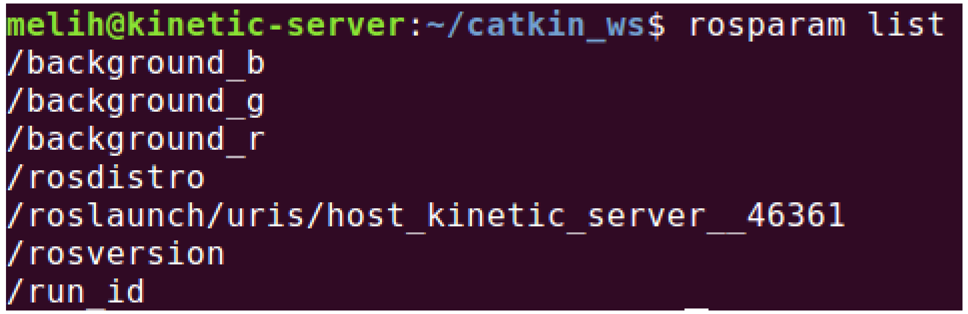


rosparam set	set parameter
rosparam get	get parameter
rosparam load	load parameters from file
rosparam dump	dump parameters to file
rosparam delete	delete parameter
rosparam list	list parameter names

Turtlesim Example

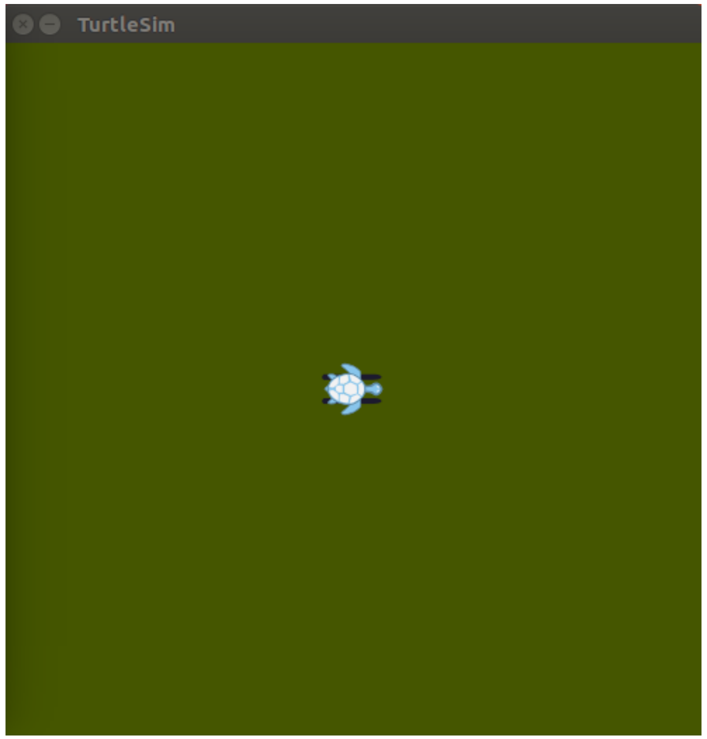
- rosrun turtlesim turtle_node

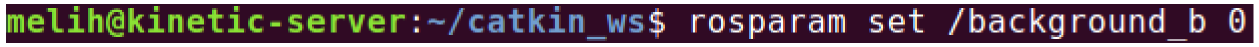




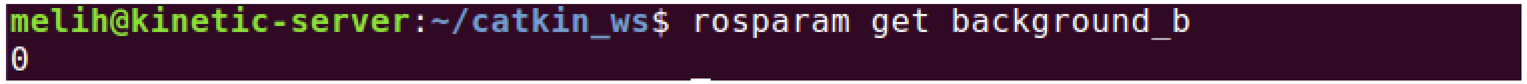
```
melih@kinetic-server:~/catkin_ws$ rosparam list
/background_b
/background_g
/background_r
/rosdistro
/roslaunch/uris/host_kinetic_server__46361
/rosversion
/run_id
```

Turtlesim Example

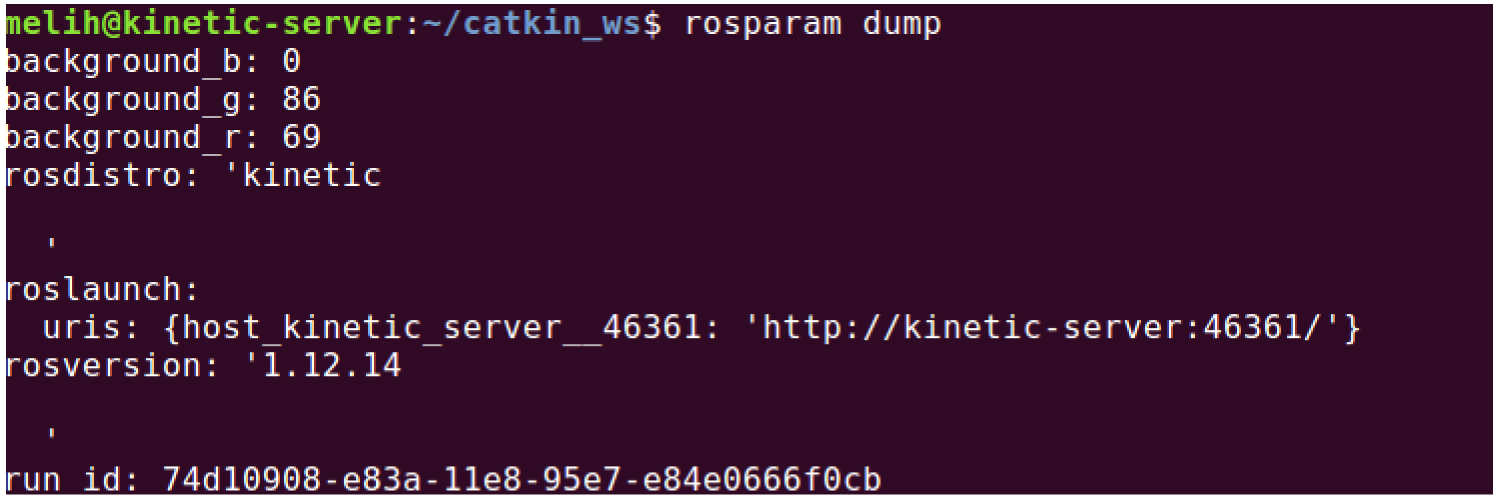




```
melih@kinetic-server:~/catkin_ws$ rosparam set /background_b 0
```

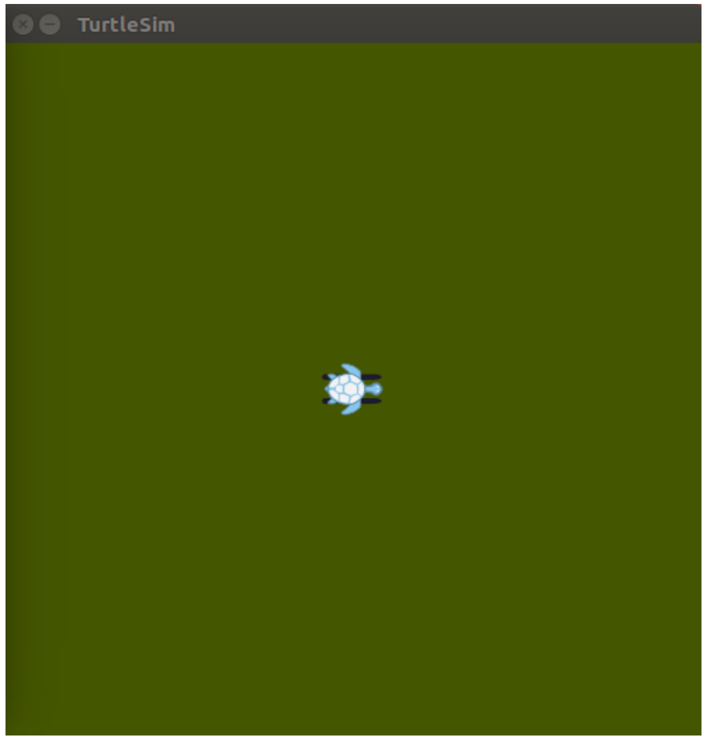


```
melih@kinetic-server:~/catkin_ws$ rosparam get background_b
0
```

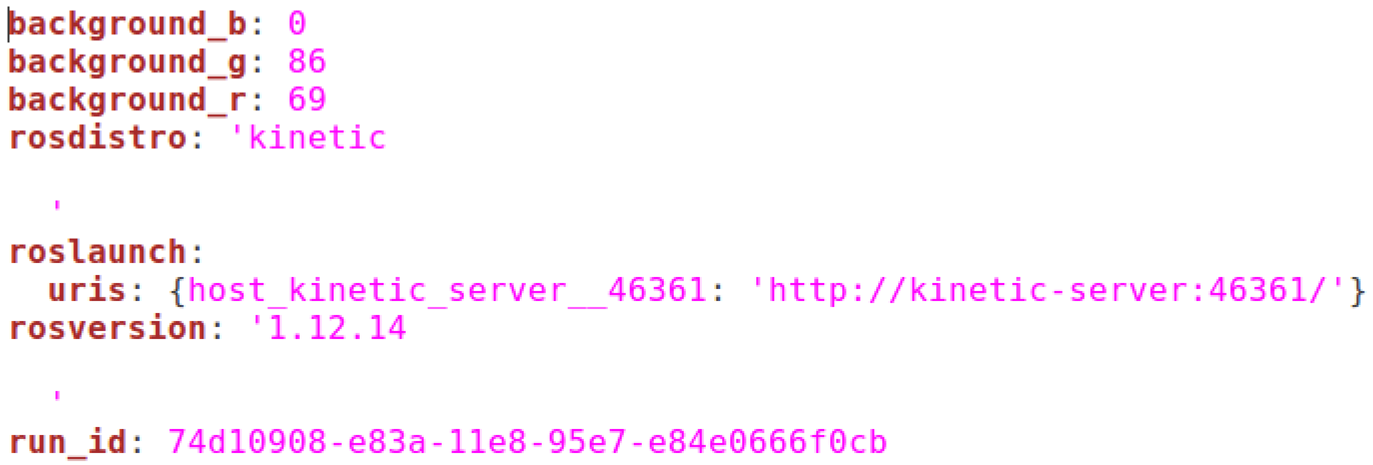


```
melih@kinetic-server:~/catkin_ws$ rosparam dump
background_b: 0
background_g: 86
background_r: 69
rosdistro: 'kinetic

  '
roslaunch:
  uris: {host_kinetic_server__46361: 'http://kinetic-server:46361/'}
rosversion: '1.12.14

  '
run_id: 74d10908-e83a-11e8-95e7-e84e0666f0cb
```

Turtlesim Example



- rosparam dump params.yaml



```
background_b: 0
background_g: 86
background_r: 69
rosdistro: 'kinetic

  '
roslaunch:
  uris: {host_kinetic_server__46361: 'http://kinetic-server:46361/'}
rosversion: '1.12.14

  '
run_id: 74d10908-e83a-11e8-95e7-e84e0666f0cb
```

- rosparam load params.yaml

ROS Parameter in Launch

- Setting a parameter value during a launch file is common practice to conveniently initialize parameters on start up.
- This can be done in your launch file using
- The <rosparam> tag enables the use of rosparam YAML files for loading and dumping parameters from the ROS Parameter Server.

```
<rosparam command="load" file="$(find rosparam)/example.yaml" />
```

```
<rosparam>
  a: 1
  b: 2
</rosparam>
```

Ros Parameter in Launch Example

```
<launch>
  <!-- set a global parameter -->
  <param name="cam" value="1" />

  <!-- set a group of parameters with parents-->
  <group ns="cameras">

    <group ns="cameraL">
      <param name="name" value="left" />
      <param name="id" value="0" />
    </group>

    <group ns="cameraR">
      <param name="name" value="right" />
      <param name="id" value="1" />
    </group>

  </group>

  <!-- set a private parameter -->
  <node pkg="tutorials" name="param" type="param.py" output="screen">
    <param name="private_param" value="secret" />
  </node>

</launch>
```

Portland State
UNIVERSITY

ROS Parameter with Rospy

- It is often the case that your nodes will have to access the parameter server during start up to retrieve configuration information, or set a parameter value.
- You can access to global and private parameters

```
# get a global parameter
rospy.get_param('/global_param_name')

# get a parameter from our parent namespace
rospy.get_param('param_name')

# get a parameter from our private namespace
rospy.get_param('~private_param_name')
```

- You can have default value for the parameter if it doesn't exist

```
rospy.get_param('foo', 'default_value')
```

ROS Parameter with Rospy

- You can delete parameters

```
rospy.delete_param('param_name')
```

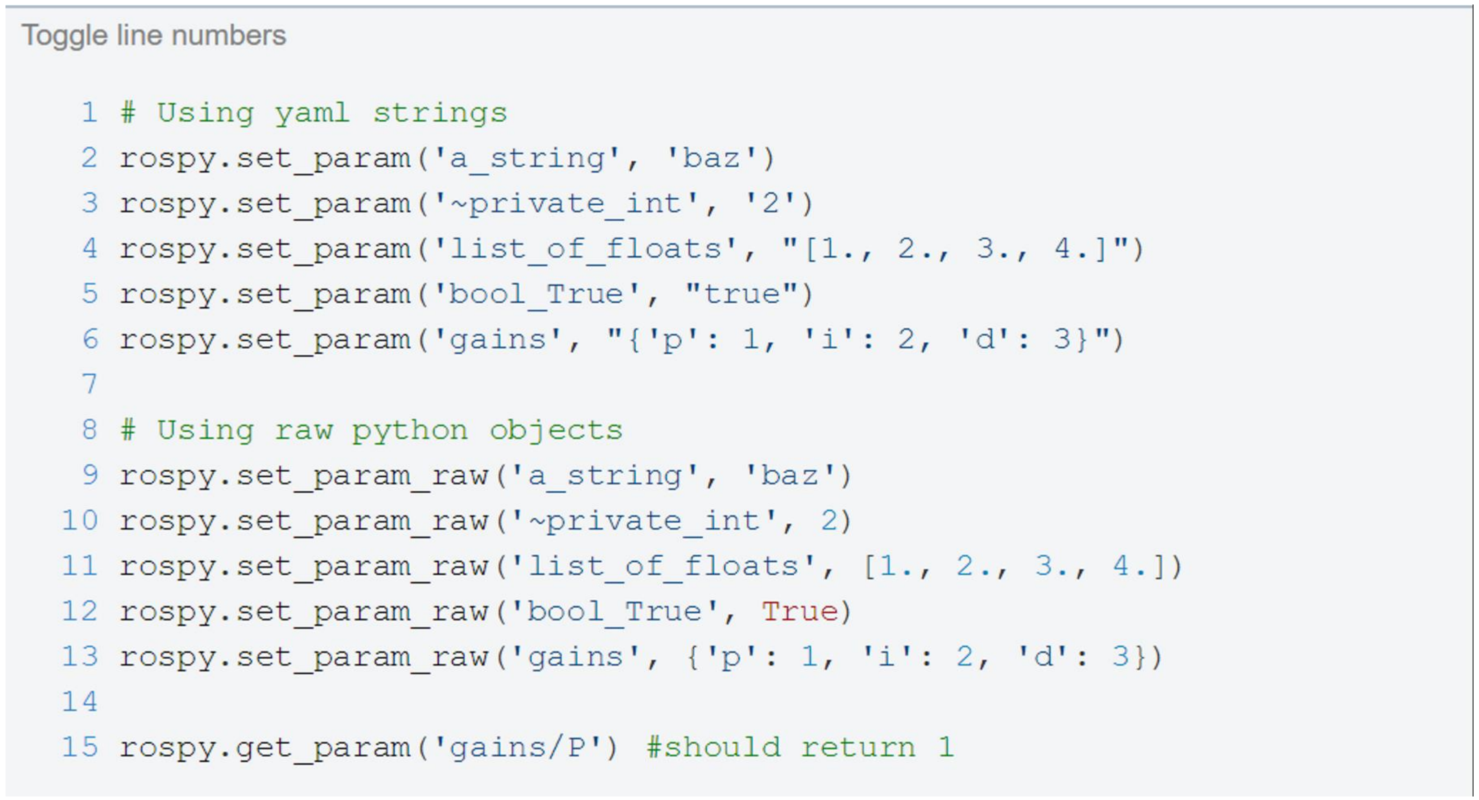
- You can check if a paramater exist

```
if rospy.has_param('to_delete'):
    rospy.delete_param('to_delete')
```

ROS Parameter with Rospy Example

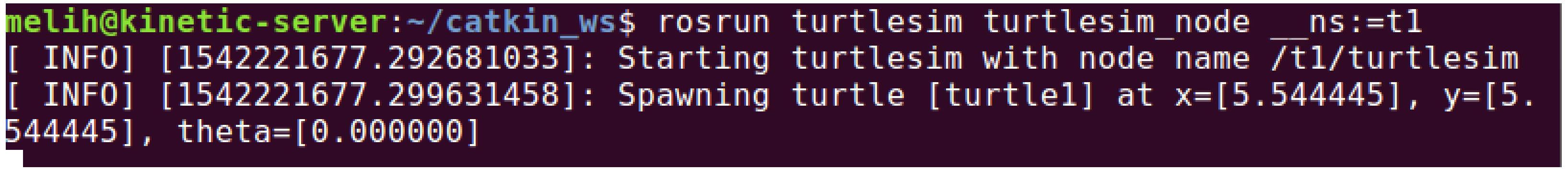
```python
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def param_talker():
    rospy.init_node('param')

    global_example = rospy.get_param("/cam")
    print (global_example)

    # fetch the group parameter from our parent namespace
    group = rospy.get_param('cameras/cameraL')
    print(group)

    # fetch a group (dictionary) of parameters
    caminfo = rospy.get_param('cameras/cameraR')
    name, camId = caminfo['name'], caminfo['id']
    rospy.loginfo("cam info are %s, %s", name, camId)

    # fetch topic_name from the ~private namespace
    private = rospy.get_param('~private_param')
    print (private)

    # search for a parameter
    param_name = rospy.search_param('private_param')
    rospy.loginfo('found it under key: %s'%param_name)

    while not rospy.is_shutdown():
        rospy.sleep(1)

if __name__ == '__main__':
    try:
        param_talker()
    except rospy.ROSInterruptException: pass
```

Parameter Examples

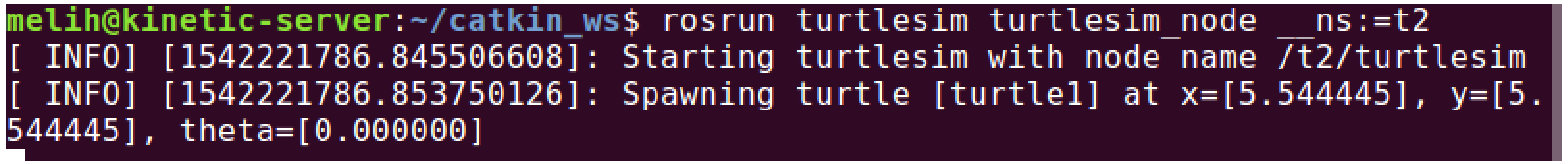


```python
# Using yaml strings
rospy.set_param('a_string', 'baz')
rospy.set_param('~private_int', '2')
rospy.set_param('list_of_floats', "[1., 2., 3., 4.]")
rospy.set_param('bool_True', "true")
rospy.set_param('gains', "{'p': 1, 'i': 2, 'd': 3}")

# Using raw python objects
rospy.set_param_raw('a_string', 'baz')
rospy.set_param_raw('~private_int', 2)
rospy.set_param_raw('list_of_floats', [1., 2., 3., 4.])
rospy.set_param_raw('bool_True', True)
rospy.set_param_raw('gains', {'p': 1, 'i': 2, 'd': 3})

rospy.get_param('gains/P') #should return 1
```

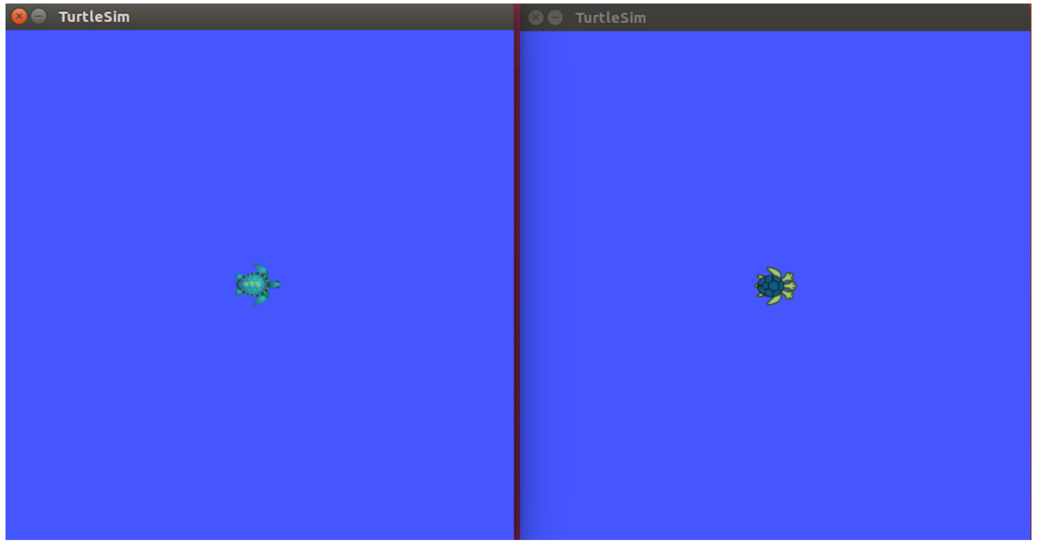
ROS Namespace



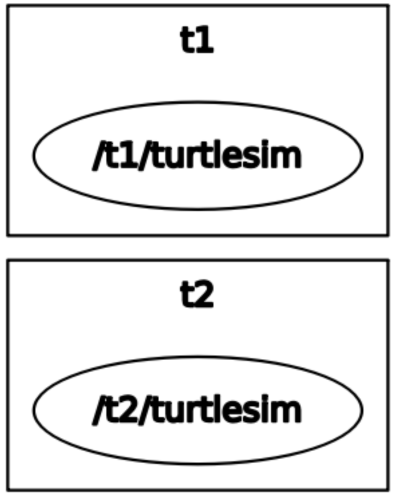
```
melih@kinetic-server:~/catkin_ws$ rosrun turtlesim turtlesim_node __ns:=t1
[ INFO] [1542221677.292681033]: Starting turtlesim with node name /t1/turtlesim
[ INFO] [1542221677.299631458]: Spawning turtle [turtle1] at x=[5.544445], y=[5.
544445], theta=[0.000000]
```

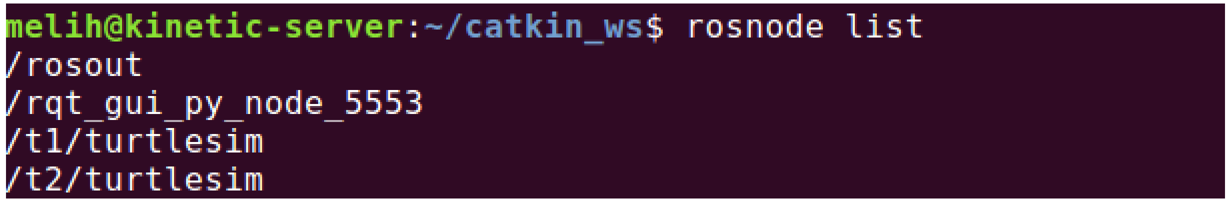


```
melih@kinetic-server:~/catkin_ws$ rosrun turtlesim turtlesim_node __ns:=t2
[ INFO] [1542221786.845506608]: Starting turtlesim with node name /t2/turtlesim
[ INFO] [1542221786.853750126]: Spawning turtle [turtle1] at x=[5.544445], y=[5.
544445], theta=[0.000000]
```

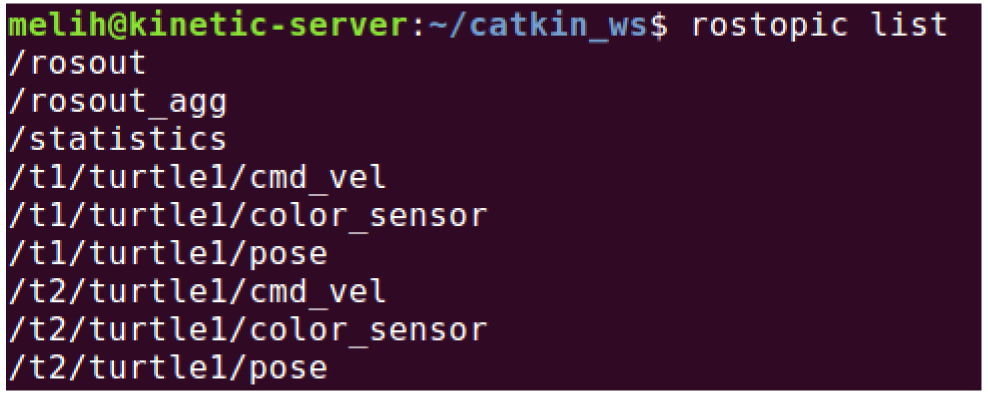


ROS Namespace





```
melih@kinetic-server:~/catkin_ws$ rosnode list
/rosout
/rqt_gui_py_node_5553
/t1/turtlesim
/t2/turtlesim
```



```
melih@kinetic-server:~/catkin_ws$ rostopic list
/rosout
/rosout_agg
/statistics
/t1/turtle1/cmd_vel
/t1/turtle1/color_sensor
/t1/turtle1/pose
/t2/turtle1/cmd_vel
/t2/turtle1/color_sensor
/t2/turtle1/pose
```

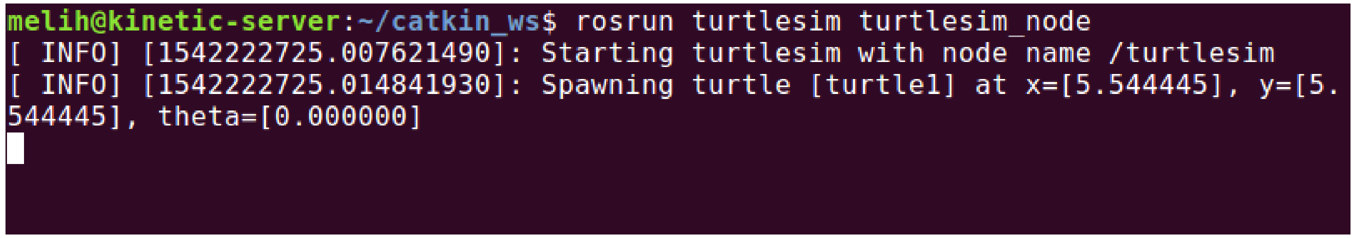
$$\underbrace{\text{/turtle1}}_{\substack{\text{default}\\\text{namespace}}} + \underbrace{\text{cmd\_vel}}_{\text{relative name}} \Rightarrow \underbrace{\text{/turtle1/cmd\_vel}}_{\text{global name}}$$

rosbag

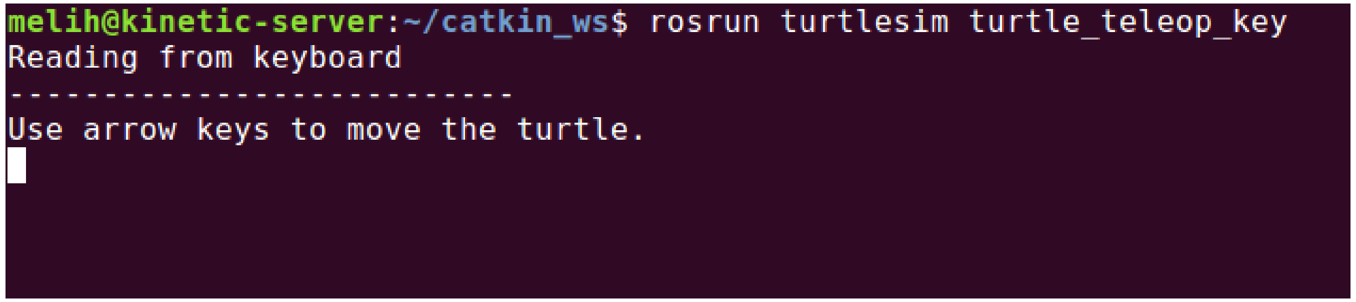
- Bags are typically created by a tool like rosbag.
- rosbag is console tool for recording, playback, and other operations.
- rosbag subscribes to one or more ROS topics, and store the serialized message data in a file as it is received.
- Rosbag files can also be played back in ROS to the same topics they were recorded from, or even remapped to new topics.

Portland State UNIVERSITY

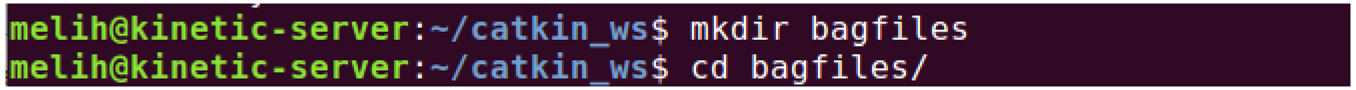
rosbag



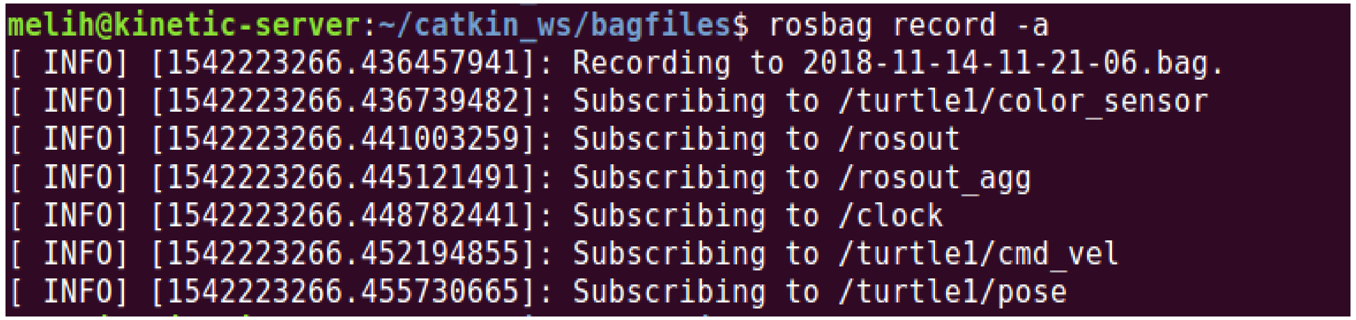
```
melih@kinetic-server:~/catkin_ws$ rosrun turtlesim turtlesim_node
[ INFO] [1542222725.007621490]: Starting turtlesim with node name /turtlesim
[ INFO] [1542222725.014841930]: Spawning turtle [turtle1] at x=[5.544445], y=[5.
544445], theta=[0.000000]
```



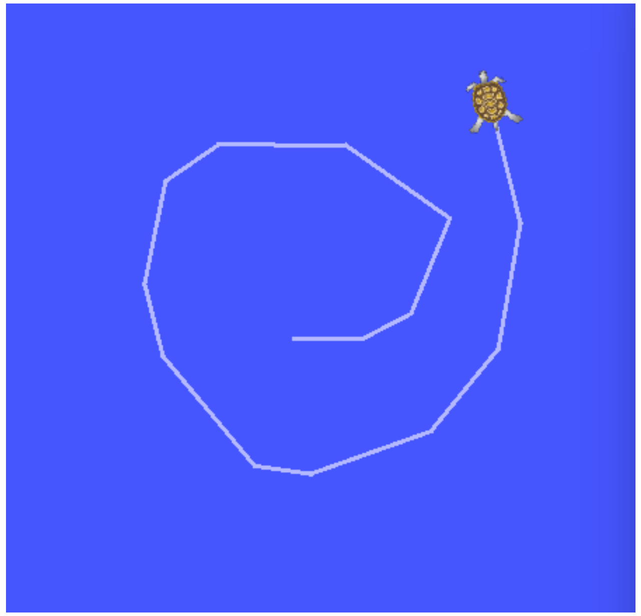
```
melih@kinetic-server:~/catkin_ws$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```



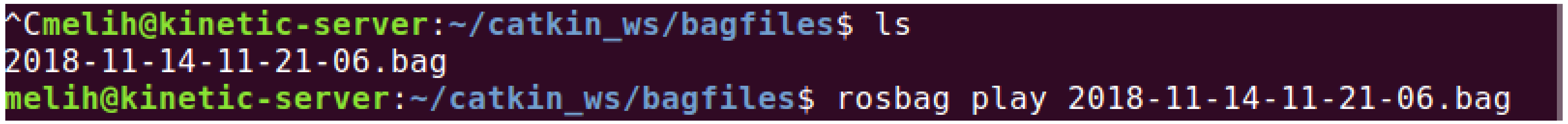
```
melih@kinetic-server:~/catkin_ws$ mkdir bagfiles
melih@kinetic-server:~/catkin_ws$ cd bagfiles/
```



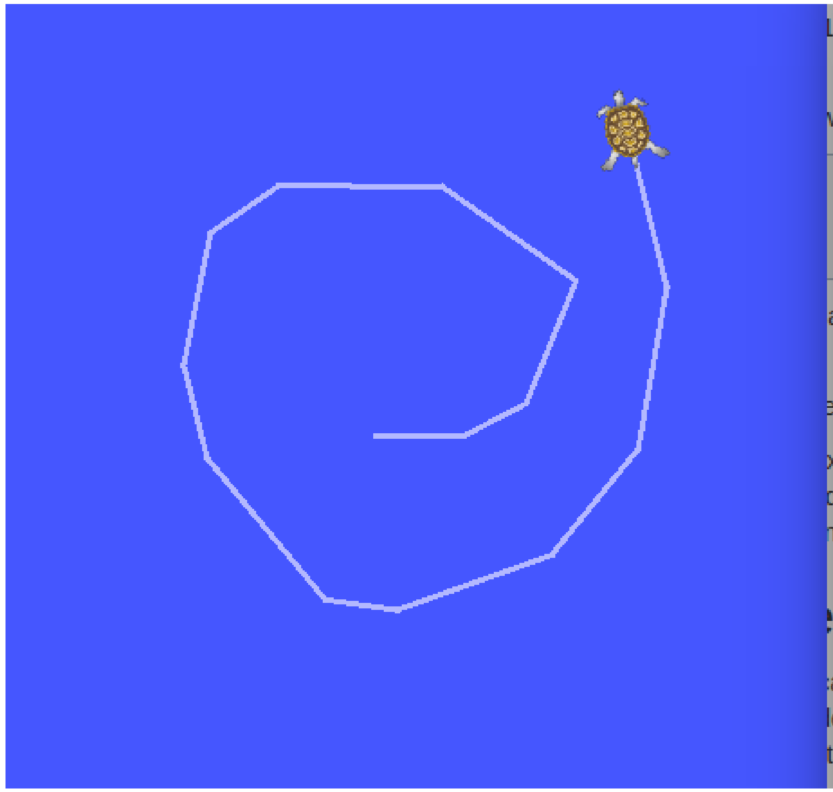
```
melih@kinetic-server:~/catkin_ws/bagfiles$ rosbag record -a
[ INFO] [1542223266.436457941]: Recording to 2018-11-14-11-21-06.bag.
[ INFO] [1542223266.436739482]: Subscribing to /turtle1/color_sensor
[ INFO] [1542223266.441003259]: Subscribing to /rosout
[ INFO] [1542223266.445121491]: Subscribing to /rosout_agg
[ INFO] [1542223266.448782441]: Subscribing to /clock
[ INFO] [1542223266.452194855]: Subscribing to /turtle1/cmd_vel
[ INFO] [1542223266.455730665]: Subscribing to /turtle1/pose
```

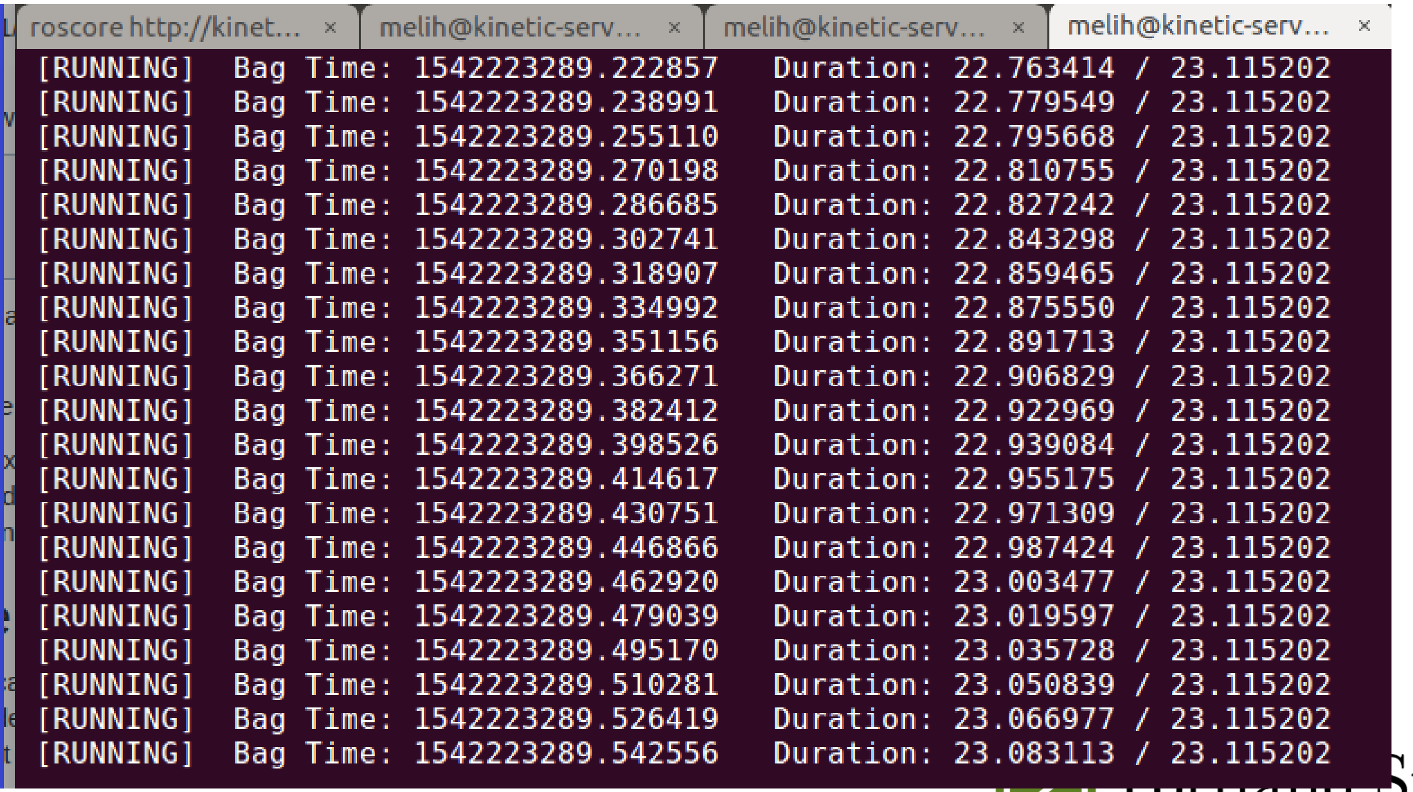


rosbag



```
^Cmelih@kinetic-server:~/catkin_ws/bagfiles$ ls
2018-11-14-11-21-06.bag
melih@kinetic-server:~/catkin_ws/bagfiles$ rosbag play 2018-11-14-11-21-06.bag
```

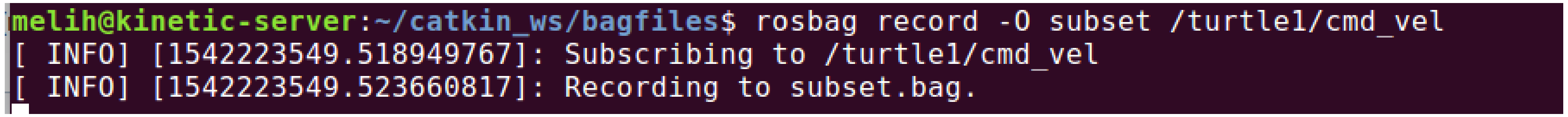




```
[RUNNING]  Bag Time: 1542223289.222857   Duration: 22.763414 / 23.115202
[RUNNING]  Bag Time: 1542223289.238991   Duration: 22.779549 / 23.115202
[RUNNING]  Bag Time: 1542223289.255110   Duration: 22.795668 / 23.115202
[RUNNING]  Bag Time: 1542223289.270198   Duration: 22.810755 / 23.115202
[RUNNING]  Bag Time: 1542223289.286685   Duration: 22.827242 / 23.115202
[RUNNING]  Bag Time: 1542223289.302741   Duration: 22.843298 / 23.115202
[RUNNING]  Bag Time: 1542223289.318907   Duration: 22.859465 / 23.115202
[RUNNING]  Bag Time: 1542223289.334992   Duration: 22.875550 / 23.115202
[RUNNING]  Bag Time: 1542223289.351156   Duration: 22.891713 / 23.115202
[RUNNING]  Bag Time: 1542223289.366271   Duration: 22.906829 / 23.115202
[RUNNING]  Bag Time: 1542223289.382412   Duration: 22.922969 / 23.115202
[RUNNING]  Bag Time: 1542223289.398526   Duration: 22.939084 / 23.115202
[RUNNING]  Bag Time: 1542223289.414617   Duration: 22.955175 / 23.115202
[RUNNING]  Bag Time: 1542223289.430751   Duration: 22.971309 / 23.115202
[RUNNING]  Bag Time: 1542223289.446866   Duration: 22.987424 / 23.115202
[RUNNING]  Bag Time: 1542223289.462920   Duration: 23.003477 / 23.115202
[RUNNING]  Bag Time: 1542223289.479039   Duration: 23.019597 / 23.115202
[RUNNING]  Bag Time: 1542223289.495170   Duration: 23.035728 / 23.115202
[RUNNING]  Bag Time: 1542223289.510281   Duration: 23.050839 / 23.115202
[RUNNING]  Bag Time: 1542223289.526419   Duration: 23.066977 / 23.115202
[RUNNING]  Bag Time: 1542223289.542556   Duration: 23.083113 / 23.115202
```

rosbag

- You don’t always want to record all the topics.
- We can record only the topics that we need.



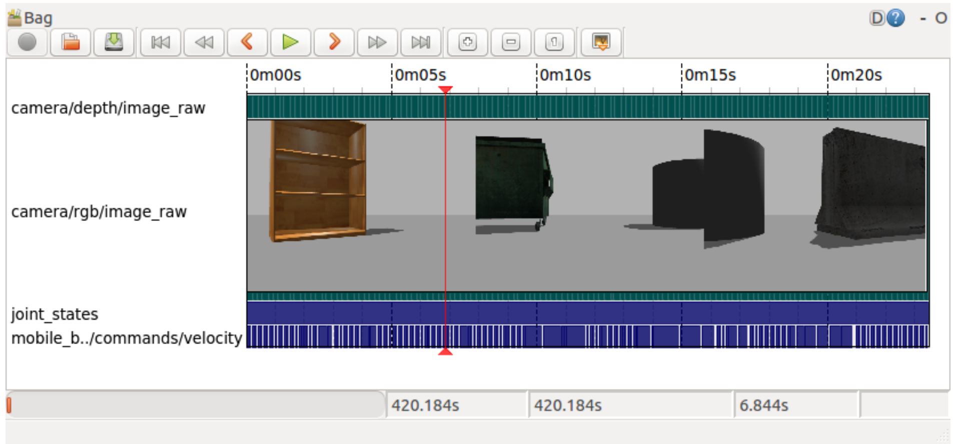
```
melih@kinetic-server:~/catkin_ws/bagfiles$ rosbag record -O subset /turtle1/cmd_vel
[ INFO] [1542223549.518949767]: Subscribing to /turtle1/cmd_vel
[ INFO] [1542223549.523660817]: Recording to subset.bag.
```

rqtbag

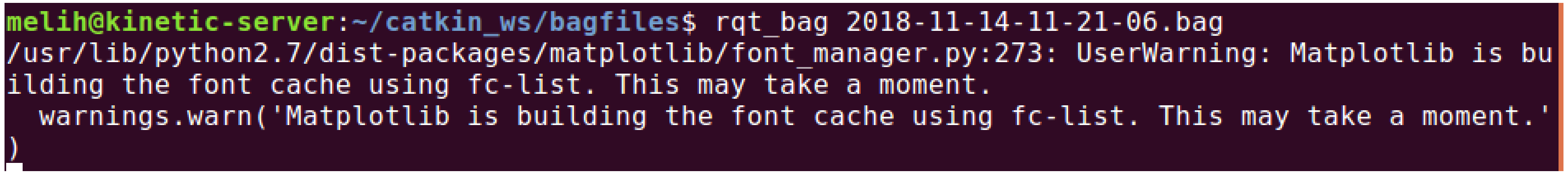
rqt_bag is an application for recording and managing bag files.

Primary features:

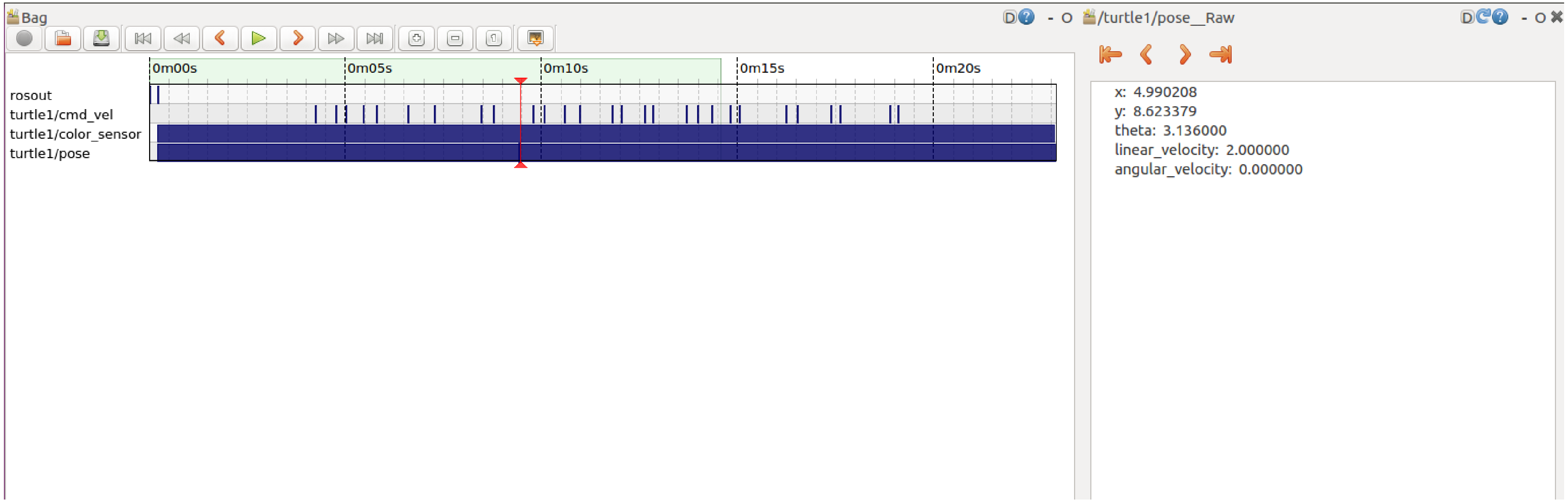
- show bag message contents
- display image messages (optionally as thumbnails on a timeline)
- plot configurable time-series of message values
- publish/record messages on selected topics to/from ROS
- export messages in a time range to a new bag
- Type rqt_bag to start it.



rqtbag

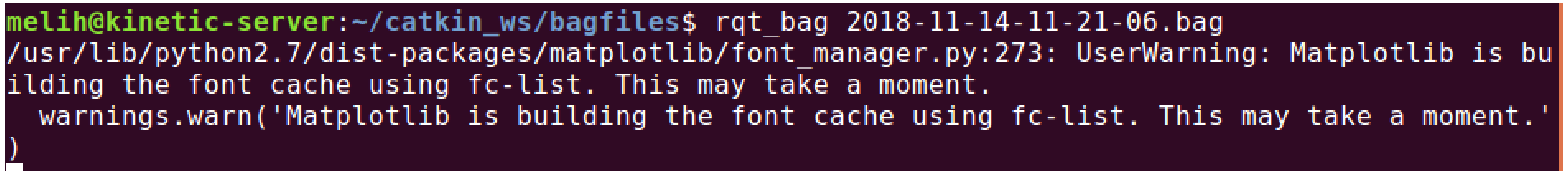


```
melih@kinetic-server:~/catkin_ws/bagfiles$ rqt_bag 2018-11-14-11-21-06.bag
/usr/lib/python2.7/dist-packages/matplotlib/font_manager.py:273: UserWarning: Matplotlib is bu
ilding the font cache using fc-list. This may take a moment.
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take a moment.'
)
```

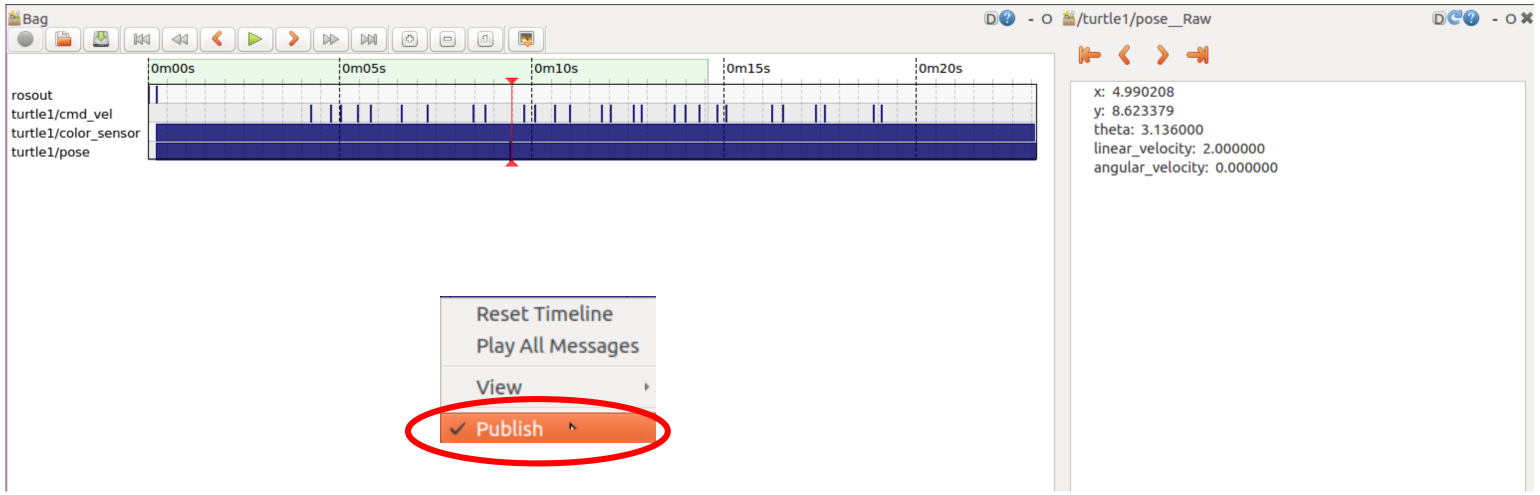


```
x: 4.990208
y: 8.623379
theta: 3.136000
linear_velocity: 2.000000
angular_velocity: 0.000000
```

rqtbag



```
melih@kinetic-server:~/catkin_ws/bagfiles$ rqt_bag 2018-11-14-11-21-06.bag
/usr/lib/python2.7/dist-packages/matplotlib/font_manager.py:273: UserWarning: Matplotlib is bu
ilding the font cache using fc-list. This may take a moment.
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take a moment.'
)
```



```
x: 4.990208
y: 8.623379
theta: 3.136000
linear_velocity: 2.000000
angular_velocity: 0.000000
```