Robot Vision Systems

Lecture 10: ROS Basics

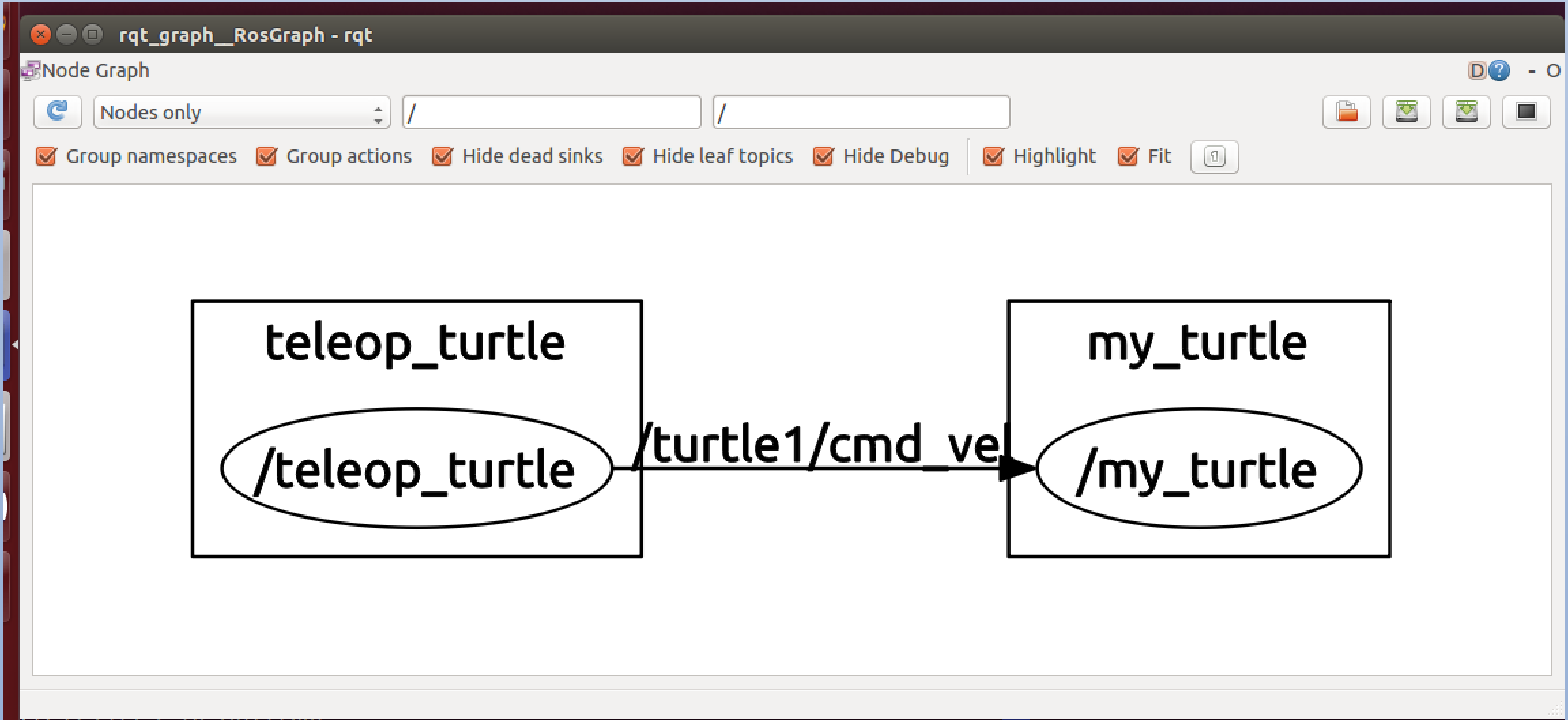
Michael Felsberg

michael.felsberg@liu.se

ROS Topics

- turtlesim_node subscribes to the same topic that turtle_teleop_key publishes to
- Visualization:

```
$ rosrun rqt_graph rqt_graph
```

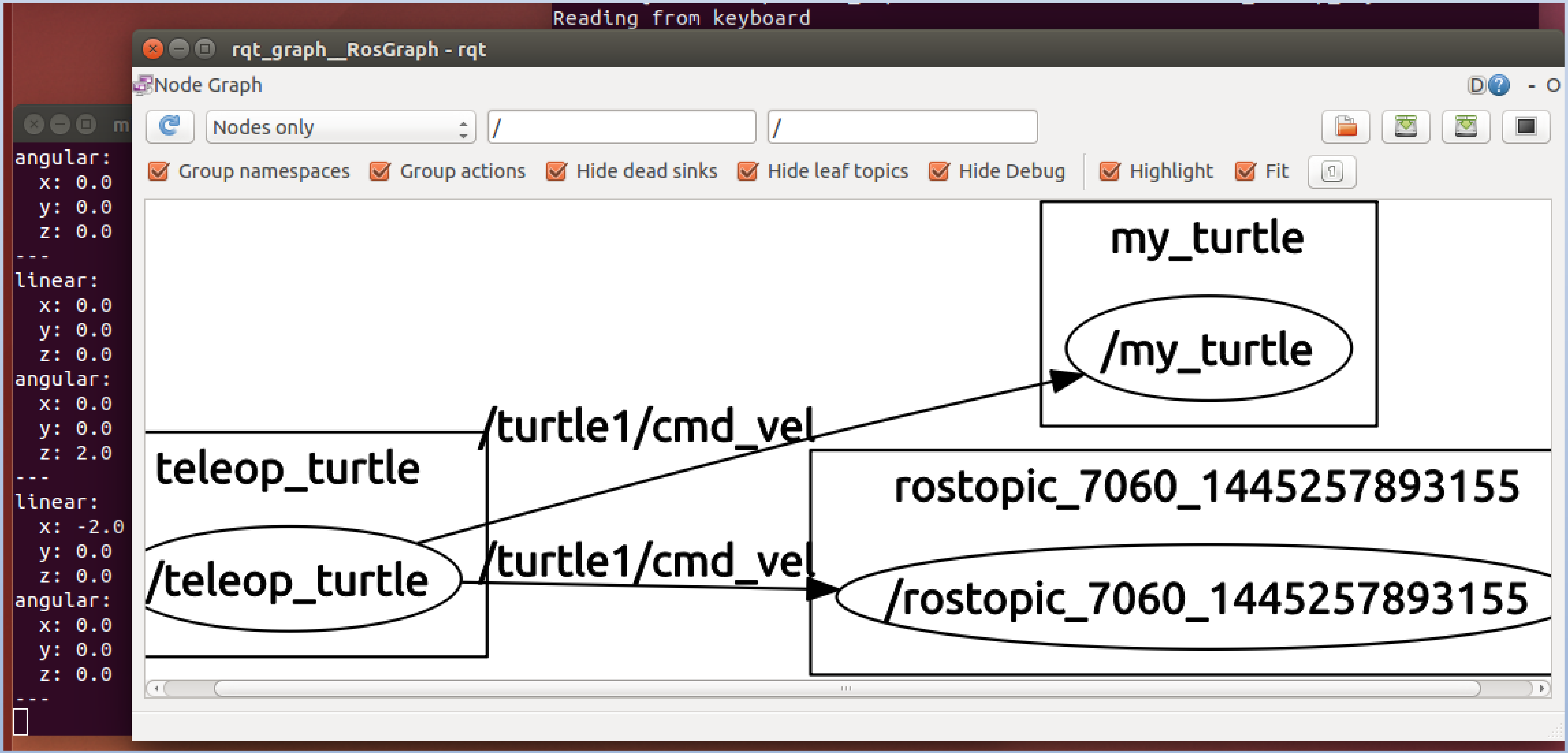


ROS Topics

- Tool: `$ 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
- Example:

```
$ rostopic echo /turtle1/cmd_vel
```

ROS Topics



```
Reading from keyboard
```

```
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 2.0
---
linear:
  x: -2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
```

ROS Topics

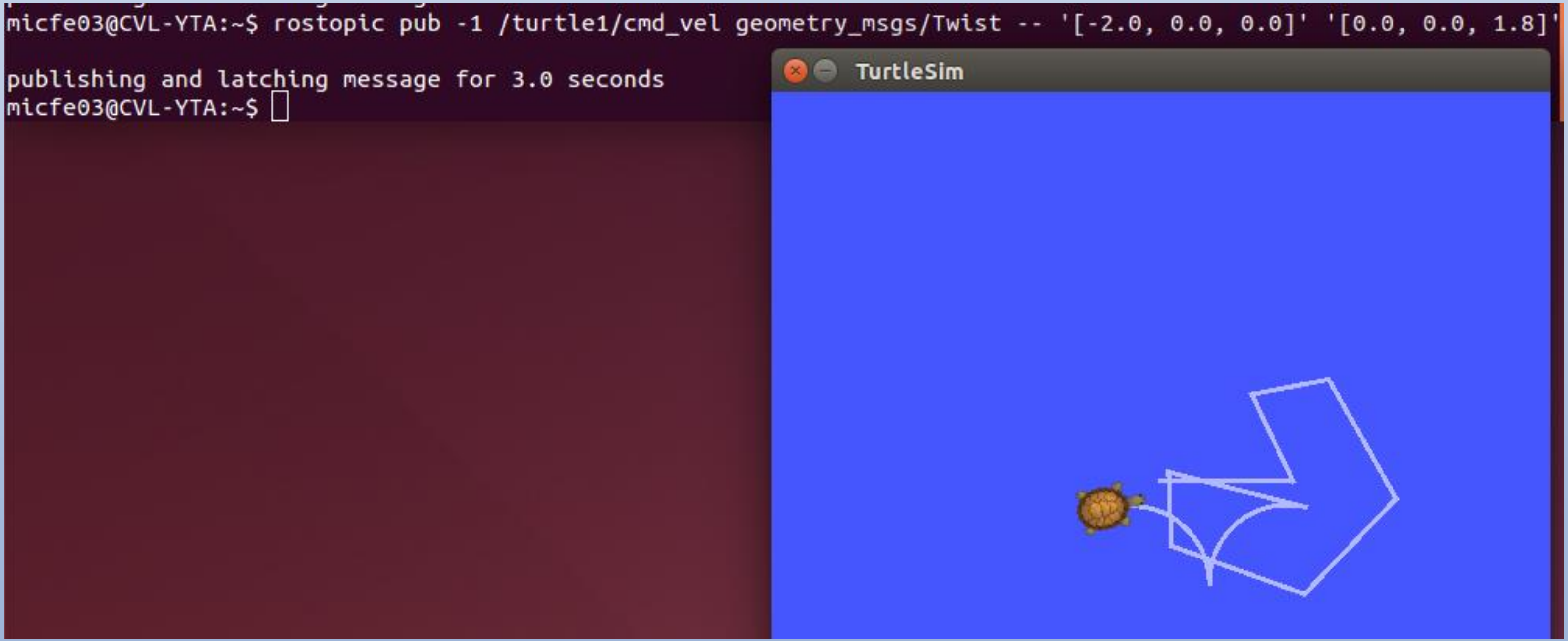
```
micfe03@CVL-YTA:~$ rostopic list -v

Published topics:
 * /turtle1/color_sensor [turtlesim/Color] 1 publisher
 * /turtle1/cmd_vel [geometry_msgs/Twist] 1 publisher
 * /rosout [rosgraph_msgs/Log] 4 publishers
 * /rosout_agg [rosgraph_msgs/Log] 1 publisher
 * /turtle1/pose [turtlesim/Pose] 1 publisher

Subscribed topics:
 * /turtle1/cmd_vel [geometry_msgs/Twist] 2 subscribers
 * /rosout [rosgraph_msgs/Log] 1 subscriber
 * /statistics [rosgraph_msgs/TopicStatistics] 1
subscriber
```

ROS Messages

```
micfe03:~$ rostopic type /turtle1/cmd_vel
geometry_msgs/Twist
micfe03:~$ 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
```

ROS Messages

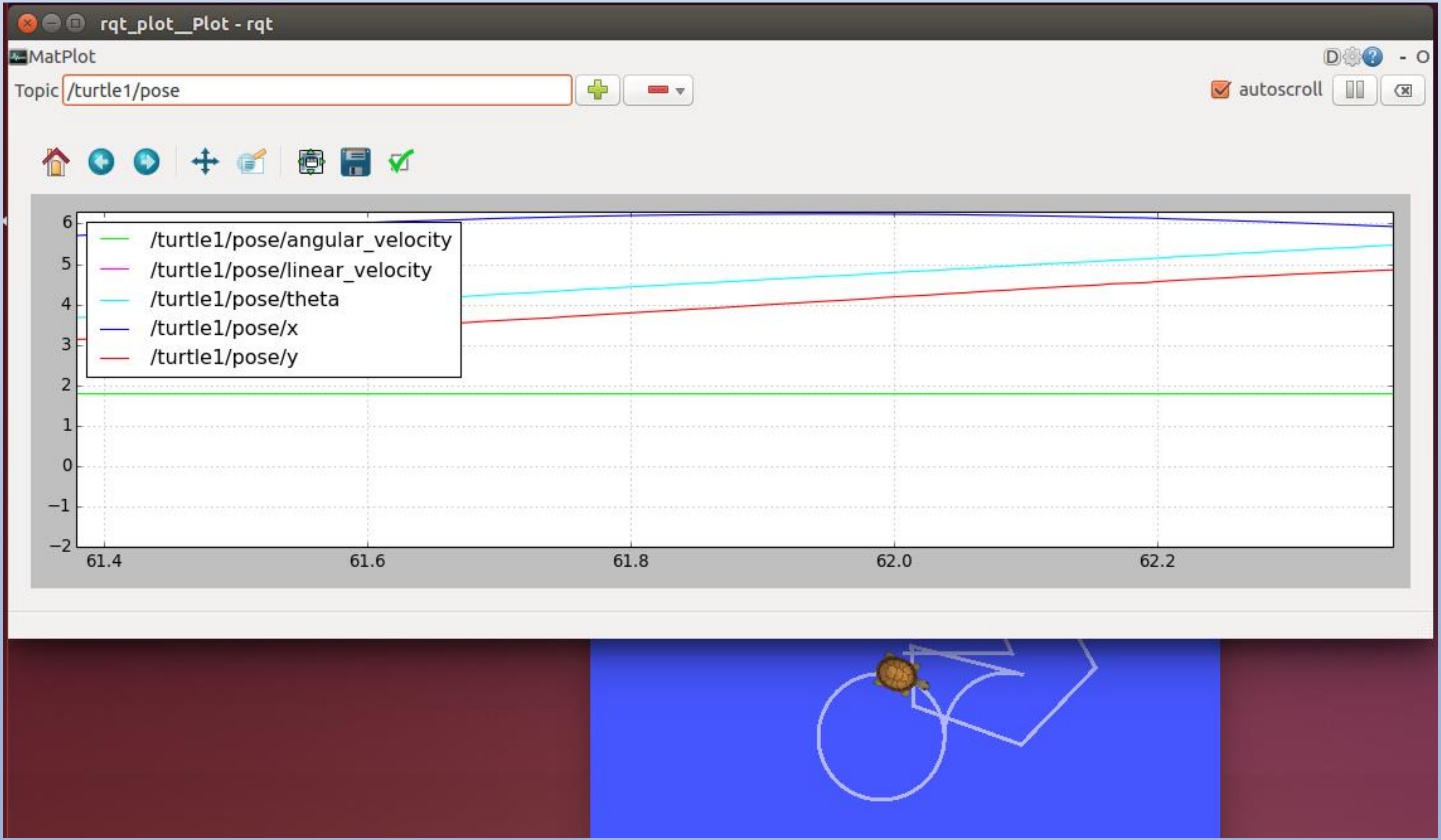
- Publish message onto topic: `$ rostopic pub [topic] [msg_type] [args]`



```
micfe03@CVL-YTA:~$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[-2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'

publishing and latching message for 3.0 seconds
micfe03@CVL-YTA:~$
```

note: `-1` vs. `-r 1`

rqt_plot


rqt_plot__Plot - rqt
MatPlot
Topic /turtle1/pose
autoscroll
/turtle1/pose/angular_velocity
/turtle1/pose/linear_velocity
/turtle1/pose/theta
/turtle1/pose/x
/turtle1/pose/y
6
5
4
3
2
1
0
−1
−2
61.4
61.6
61.8
62.0
62.2

Topics vs. Services

- Topics are published and subscribed to
  - –Streams
  - –Many to many
  - –Publisher decides when to send
  - –Callbacks receive data once available
- Services are requested by client
  - –On demand
  - –One specific task
  - –Remote procedure call
  - –One to one
  - –Actionlib option

ROS Services

- Node: send request and receive response
- Rosservice tool

```
rosservice list  print information
                 about active services
rosservice call  call the service
                 with the provided
                 args
rosservice type  print service type
rosservice find  find services by
                 service type
rosservice uri   print service
                 ROSRPC uri
```

List and Type

- Example: turtlesim provides nine services, such as reset, clear, spawn, kill, etc

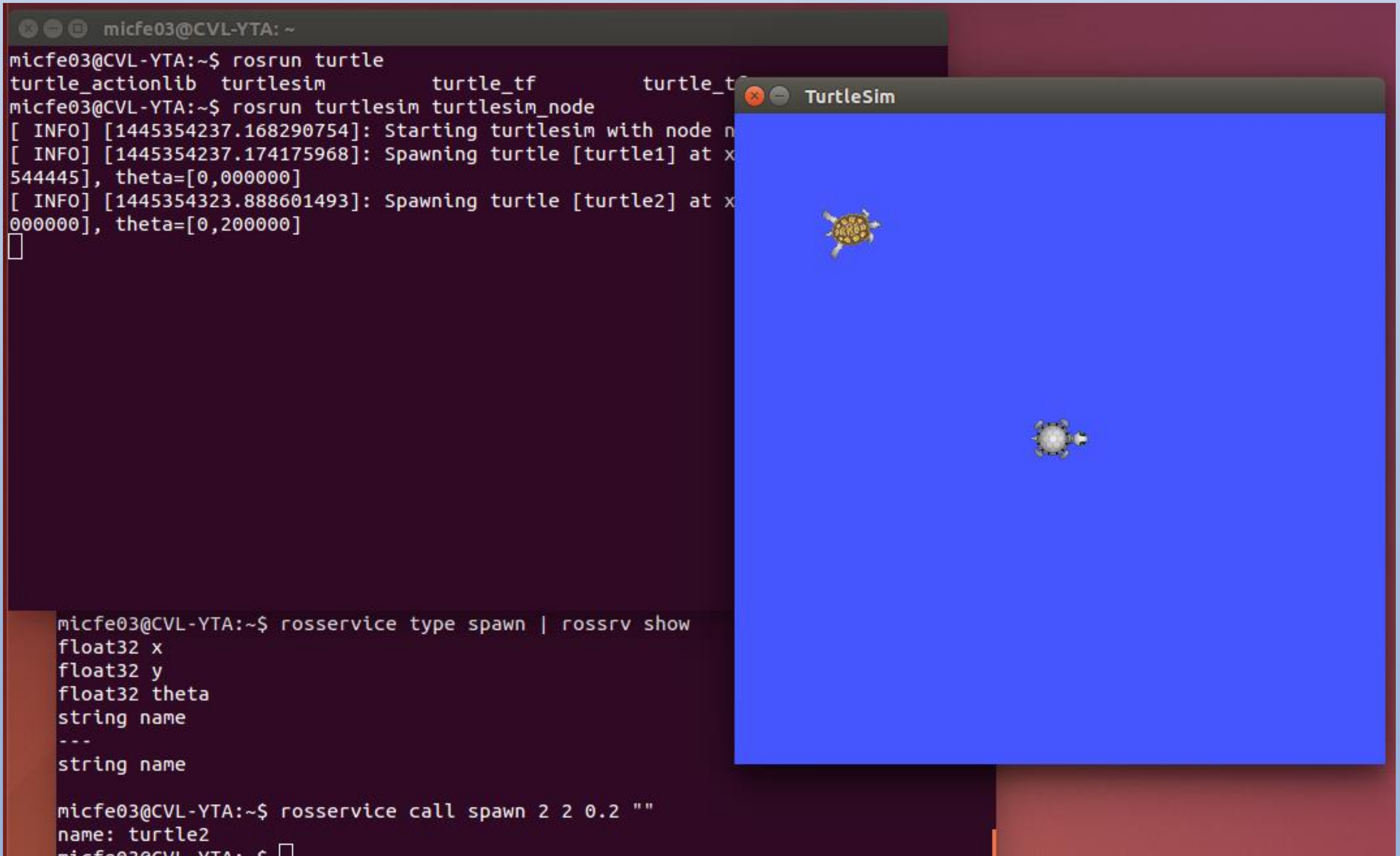
```
$ rosservice list
```

- Input/output types for services

```
rosservice type [service]
$ rosservice type spawn| rossrv show
float32 x
float32 y
float32 theta
string name
---
string name
```

Call

- If we know service and type, we can call it
  `rosservice call [service] [args]`
  `$ rosservice call spawn 2 2 0.2 ""`



```
micfe03@CVL-YTA:~$ rosrun turtle
turtle_actionlib  turtlesim        turtle_tf        turtle_t
micfe03@CVL-YTA:~$ rosrun turtlesim turtlesim_node
[ INFO] [1445354237.168290754]: Starting turtlesim with node n
[ INFO] [1445354237.174175968]: Spawning turtle [turtle1] at x
544445], theta=[0,000000]
[ INFO] [1445354323.888601493]: Spawning turtle [turtle2] at x
000000], theta=[0,200000]
```

```
micfe03@CVL-YTA:~$ rosservice type spawn | rossrv show
float32 x
float32 y
float32 theta
string name
---
string name

micfe03@CVL-YTA:~$ rosservice call spawn 2 2 0.2 ""
name: turtle2
```

ROS Parameter Server

- YAML markup language

`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

List and Set

- Turtle sim has three parameters

```
$ rosparam list
/background_b
/background_g
/background_r
```

- Modify parameter and read parameter

```
rosparam set [param_name]
```

example

```
$ rosparam set background_r 150
$ rosservice call clear
```

Get, Dump & Load

- Read parameters

  ```
  rosparam get [param_name]
  ```

  example

  ```
  $ rosparam get /
  background_b: 255
  background_g: 86
  background_r: 150
  ```

- Dump and load

  ```
  rosparam dump [file_name] [namespace]
  rosparam load [file_name] [namespace]
  ```

  example

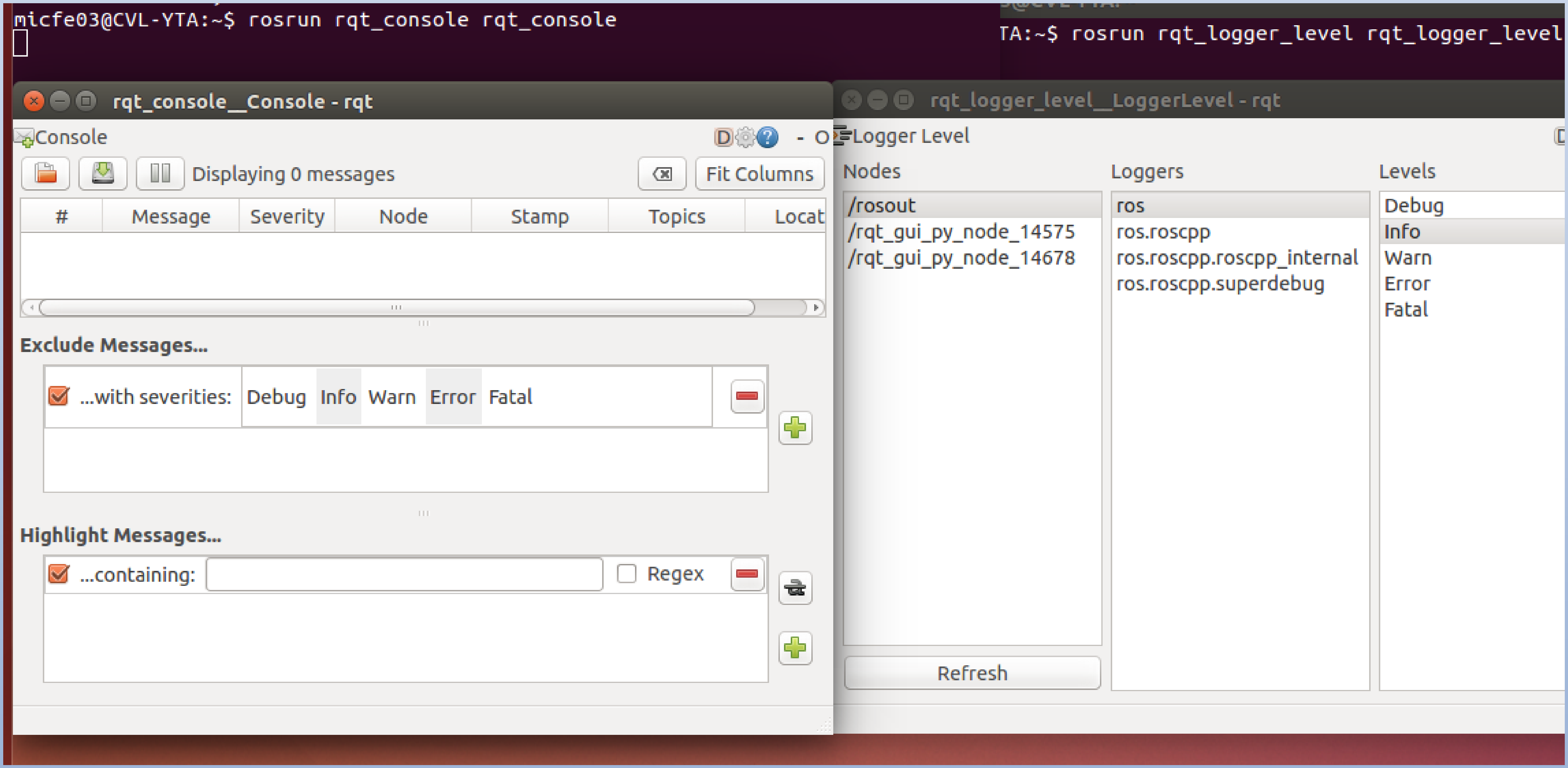
  ```
  $ rosparam dump params.yaml
  $ rosparam load params.yaml copy
  ```

Debugging Tools

- In rqt_console messages can be shown
- In rqt_logger_level the level of detail is chosen:
  - DEBUG
  - WARN
  - INFO
  - ERROR
  - FATAL
- Start in two terminals

```
$ rosrun rqt_console rqt_console
$ rosrun rqt_logger_level
                  rqt_logger_level
```

Debugging Tools



```
micfe03@CVL-YTA:~$ rosrun rqt_console rqt_console
```

```
TA:~$ rosrun rqt_logger_level rqt_logger_level
```

rqt_console__Console - rqt
Console
Displaying 0 messages
Fit Columns
# Message Severity Node Stamp Topics Locat
Exclude Messages...
...with severities: Debug Info Warn Error Fatal
Highlight Messages...
...containing:
Regex
rqt_logger_level__LoggerLevel - rqt
Logger Level
Refresh

Nodes	Loggers	Levels
/rosout	ros	Debug
/rqt_gui_py_node_14575	ros.roscpp	Info
/rqt_gui_py_node_14678	ros.roscpp.roscpp_internal	Warn
	ros.roscpp.superdebug	Error
		Fatal

Launching Nodes

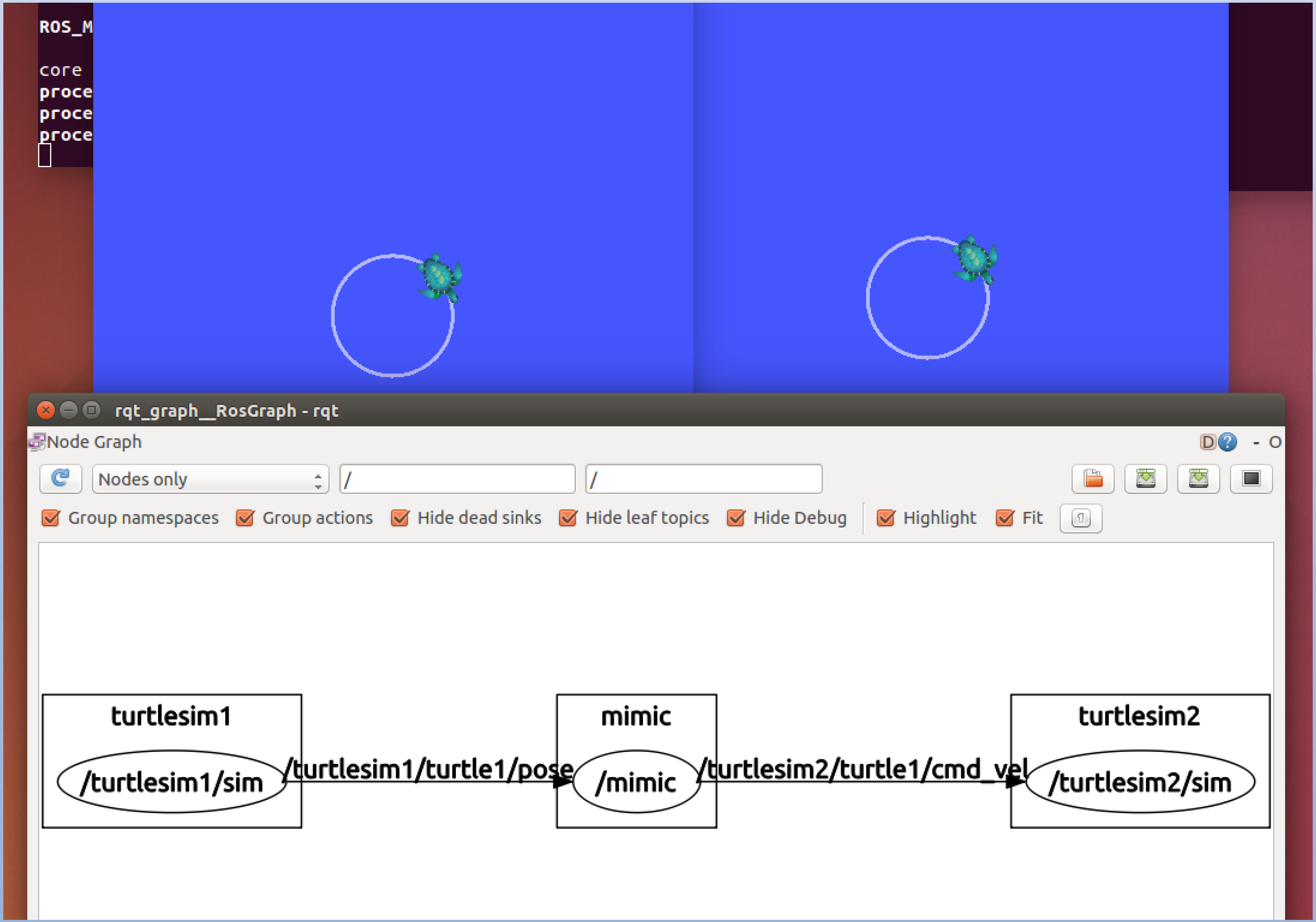
- Use roslaunch to start multiple nodes

```
$ roslaunch [package] [filename.launch]
```

- uses launch file, preferably in “launch” catalogue

```
<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>
```

Launching Nodes



```
ROS_M

core
proce
proce
proce
```

The Editor

- Check your environment `$ echo $EDITOR`
- If empty, you have to set it in your .bashrc
  `export EDITOR='nano -w`` or install vim
- The rosbash suite contains a wrapper
  ```
  $ rosed [package_name] [filename]
  ```
  example
  ```
  $ rosed roscpp <tab><tab>
  ```
- Another useful tool
  ```
  $ roscp [package_name]
  [file_to_copy_path] [copy_path]
  ```

Defining Messages

- Messages (as used in Topics) are defined in a text file: the msg file (in the msg directory)
- Used also for source code generation
- Note that several Topics may use the same message (but not vice-versa)
- Each line contains field type and name
  - int8, int16, int32, int64 (plus uint*)
  - float32, float64
  - string
  - time, duration
  - other msg files
  - variable-length array[] and fixed-length array[C]
  - Header

Example msg File

- `Header` is a special ROS type
  - Timestamp
  - Coordinate frame information
- FrameGeometry.msg

```
Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist
```

Code Generation

- Create msg file in `<package_name>/msg/`
- In the manifest package.xml uncomment

```
<build_depend>message_generation</build_depend>
<run_depend>message_runtime</run_depend>
```

- In CMakeLists.txt add

```
find_package(catkin REQUIRED
COMPONENTS
   roscpp
   rospy
   std_msgs
   message_generation
)
```

Code Generation

- Further changes in CMakeLists.txt

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

- And (to allow CMake knowing when to reconfig)

```
add_message_files(
  FILES
  FrameGeometry.msg
)
```

Code Generation

- Final change in CMakeLists.txt

```
generate_messages(
  DEPENDENCIES
  std_msgs
)
```

- Note that all these sections are already there, just commented out
- Verify msg entry

```
$ rosmsg show
[beginner_tutorials/]FrameGeometry
```

Defining Services

- By srv file in the srv directory
- A request and a response part, divided by `triple dash (---)`

```
int64 A
int64 B
---
int64 Sum
```

- Example: copy

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

Code Generation

- In the manifest package.xml uncomment (as before)

```
<build_depend>message_generation</build_depend>
<run_depend>message_runtime</run_depend>
```

- In CMakeLists.txt add (as before)

```
find_package(catkin REQUIRED COMPONENTS
   roscpp
   rospy
   std_msgs
   message_generation
)
```

Code Generation

- Despite name: message_generation for msg and srv
- Use same additional dependencies

```
CATKIN_DEPENDS, generate_messages()
```

- Add explicitly service file

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

- Check service

```
$ rossrv show
[beginner_tutorials/]AddTwoInts
```

Generate Code (msg & srv)

- Call in your catkin workspace
  ```
  $ catkin_make
  ```
- C++ header files
  ```
  ~/catkin_ws/devel/include/beginner_tuto
  rials
  ```
- Python scripts
  ```
  ~/catkin_ws/devel/lib/python2.7/dist-
  packages/beginner_tutorials/msg (…/srv)
  ```
- Lisp files
  ```
  ~/catkin_ws/devel/share/common-
  lisp/ros/beginner_tutorials/msg (…/srv)
  ```

ROS Level Debugging

- System-wide tool for analyzing the local ROS installation: `$ roswtf`

```
No package or stack in context
================================================================
Static checks summary:

No errors or warnings
================================================================
Beginning tests of your ROS graph. These may take awhile...
analyzing graph...
... done analyzing graph
running graph rules...
... done running graph rules

Online checks summary:

Found 1 warning(s).
Warnings are things that may be just fine, but are sometimes at fault

WARNING The following node subscriptions are unconnected:
 * /rosout:
   * /rosout
```

Data Recording

- ROS uses .bag files to record data from a running ROS system
- Record all topics

```
mkdir ~/bagfiles
cd ~/bagfiles
rosbag record -a
```

- Information about the .bag file

```
rosbag info <your bagfile>
```

- Play back .bag file (quit publishing nodes first)

```
rosbag play <your bagfile>
```

Data Recording

- bag files are usually named year-mo-da-ho-mi-se.bag. Change with -O
- Playback can be modified with these options:
  - –d delay before playback start (def. 0.2 s)
  - –s start playback some duration past the beginning
  - –r playback at different rate
- Recording can be limited to particular topics

```
rosbag record -O subset
/turtle1/cmd_vel /turtle1/pose
```

ROS Help

```
$ rosmsg -h
Commands:
   rosmsg show Show message description
   rosmsg users   Find files that use
                    message
   rosmsg md5   Display message md5sum
   rosmsg package   List messages in a
                    package
   rosmsg packages List packages that
                    contain messages


$ rosmsg show -h
```

Summary rosbash

- rospack = ros+pack(age) : provides information related to ROS packages
- roscd = ros+cd : changes directory to a ROS package
- rosls = ros+ls : lists files in a ROS package
- roscp = ros+cp : copies files from/to a ROS package
- rosmsg = ros+msg : provides information related to ROS message definitions
- rossrv = ros+srv : provides information related to ROS service definitions
- catkin_make : makes (compiles) a ROS package