ROS2: Other CLI tools

운영체제의 실제

안인규 (Inkyu An)

What is RQT?

- **Qt-based GUI tools**
  - **rqt_graph**: visualizes the nodes and topic flow as a graph
  - **rqt_topic**: checks the topic list, type, publishing rate, and bandwidth
  - **rqt_publisher**: publishes message via a GUI after selecting the message type and inputting values
  - **rqt_plot**: plot numeric fields in a real-time graph (e.g., /turtle1/pose/x)
  - **rqt_service_caller**: sends service requests through a GUI
  - **rqt_tf_tree**: check the frame tree as a tree/graph (We will see this later)

Recap: Nodes

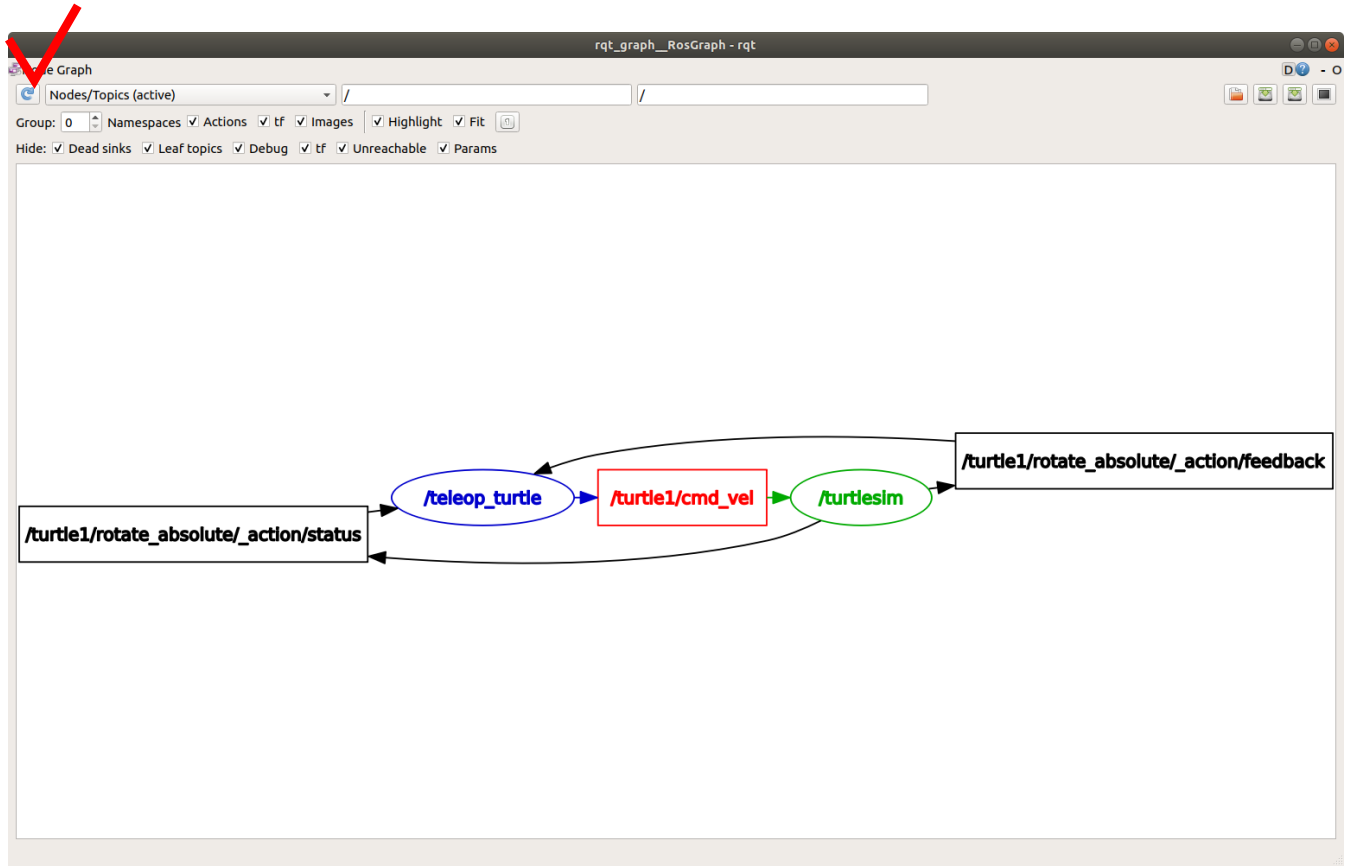
- The command ‘ros2 run’ launches an executable from a package
  - *ros2 run <package_name> <executable_name>*
  - e.g., ros2 *run turtlesim turtlesim_node*

- ‘ros2 node list’ will show you the names of all running nodes
  - *ros2 node list*

- Open another new terminal and start the teleop node with the c ommands:
  - *ros2 run turtlesim turtle_teleop_key*

What is RQT?

- *rqt_graph* visualizes node and topic flow as a graph
  - *ros2 run rqt_graph rqt_graph*

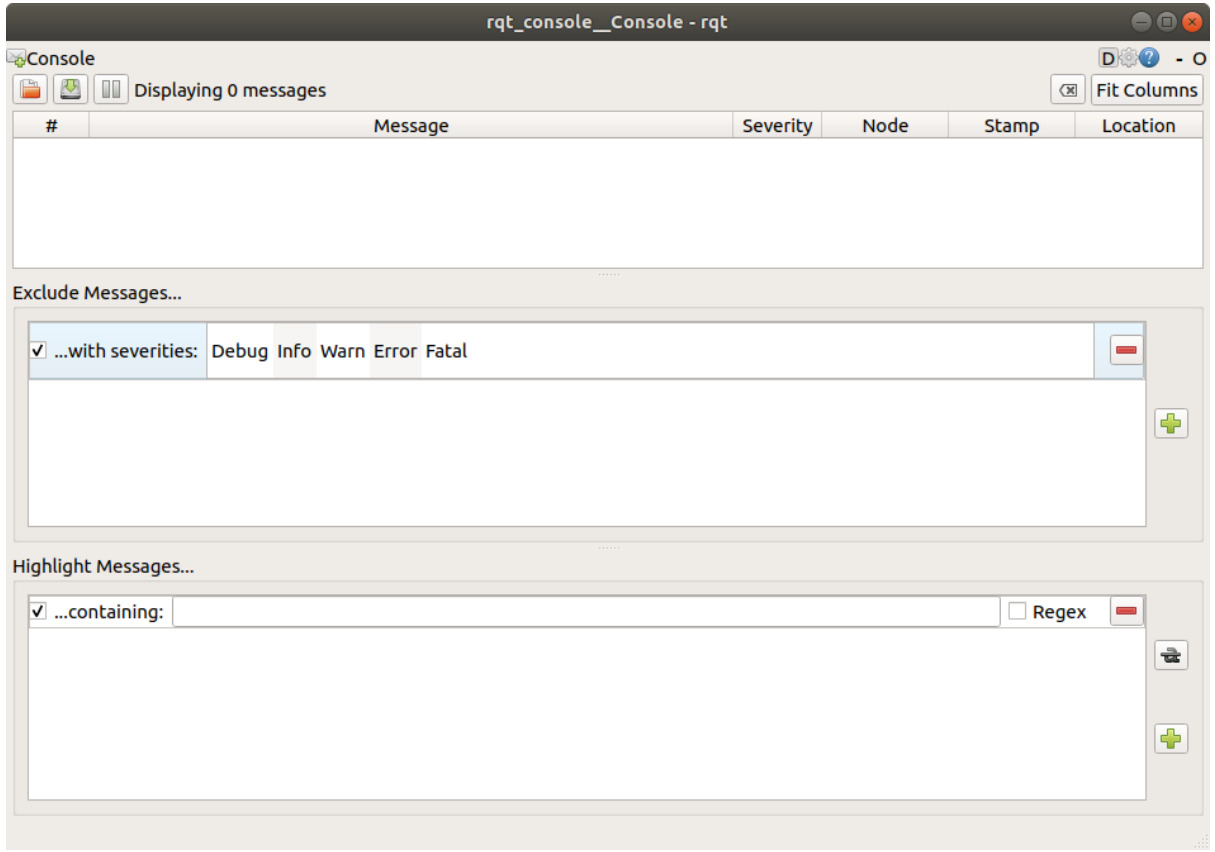


What is RQT? | Practice

- **Qt-based GUI tools**
  - **rqt_topic**: checks the topic list, type, publishing rate, and bandwidth
  - **rqt_publisher**: publishes message via a GUI after selecting the message type and inputting values
  - **rqt_plot**: plot numeric fields in a real-time graph (e.g., /turtle1/pose/x)

What is RQT?

- Using *rqt_console* to view logs
  - *ros2 run rqt_console rqt_console*



Log messages from your system will display

The option to filter messages by excluding severity levels

Highlighting messages that include a string you input

What is RQT?

- ROS2's logger levels are ordered by severity:
  1. **Fatal** messages indicate the system is going to terminate to try to pro tect itself from detriment.
  2. **Error** messages indicate significant issues that won't necessarily dama ge the system, but are preventing it from functioning properly.
  3. **Warn** messages indicate unexpected activity or non-ideal results that might represent a deeper issue, but don't harm functionality outright.
  4. **Info** messages indicate event and status updates that serve as a visual verification that the system is running as expected.
  5. **Debug** messages detail the entire step-by-step process of the system execution.

What is RQT?

- ROS2's logger levels are ordered by severity:
  1. **Fatal** messages indicate the system is going to terminate to try to pro tect itself from detriment.
  2. **Error** messages indicate significant issues that won't necessarily dama ge the system, but are preventing it from functioning properly.
  3. **Warn** messages indicate unexpected activity or non-ideal results that might represent a deeper issue, but don't harm functionality outright.
  4. **Info** messages indicate event and status updates that serve as a visual verification that the system is running as expected.
  5. **Debug** messages detail the entire step-by-step process of the system execution.

The default level is Info! → **What is the meaning?**

What is RQT?

- Set the default logger level when you first run the node
  - *ros2 run turtlesim turtlesim_node --ros-args --log-level WARN*

Launching nodes

- We have been opening new terminals for every new node you run
- As you create more complex systems with more and more nodes running simultaneously, opening terminals and reentering configuration details becomes tedious
- **Launch files** allow you to start up and configure a number of executables containing ROS 2 nodes simultaneously

Launching nodes

- Open a new terminal and run:

```
$ ros2 launch turtlesim multisim.launch.py
```

```python
from launch import LaunchDescription
import launch_ros.actions


def generate_launch_description():
    return LaunchDescription([
        launch_ros.actions.Node(
            namespace='turtlesim1', package='turtlesim',
            executable='turtlesim_node', output='screen'),
        launch_ros.actions.Node(
            namespace='turtlesim2', package='turtlesim',
            executable='turtlesim_node', output='screen'),
    ])
```

Launching nodes

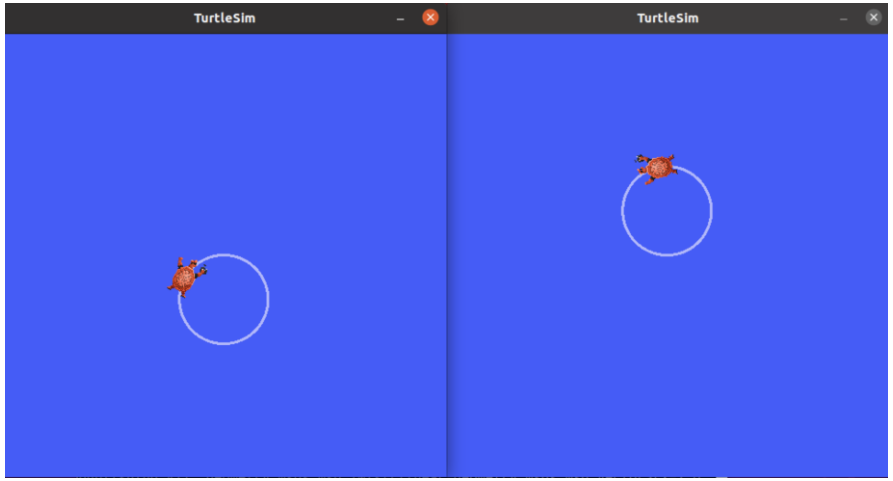
- Open a new terminal and run:

```
$ ros2 launch turtlesim multisim.launch.py
```

```python
from launch import LaunchDescription
import launch_ros.actions


def generate_launch_description():
    return LaunchDescription([
        launch_ros.actions.Node(
            namespace='turtlesim1', package='turtlesim',
            executable='turtlesim_node', output='screen'),
        launch_ros.actions.Node(
            namespace='turtlesim2', package='turtlesim',
            executable='turtlesim_node', output='screen'),
    ])
```

Q: How to control both turtles? Make them draw circles!



Launching nodes

- Option 1: utilizing the CLI command (ros2 topic pub ...)
  - *ros2 topic pub /turtlesim1/turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"*
  - *ros2 topic pub /turtlesim2/turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: -1.8}}"*
- Option 2: utilizing the RQT tool
  - *ros2 run rqt_publisher rqt_publisher*

Recording and playing back data

- **ros2 bag** is a command line tool for recording data published on topics in your system
- It accumulates the data passed on any number of **topics** and saves it in a database
- We can then replay the data to reproduce the results of your tests and experiments
- Recording topics is also a great way to share your work and allow others to recreate it.

Recap: Nodes

- The command ‘ros2 run’ launches an executable from a package
  - *ros2 run <package_name> <executable_name>*
  - e.g., ros2 *run turtlesim turtlesim_node*

- ‘ros2 node list’ will show you the names of all running nodes
  - *ros2 node list*

- Open another new terminal and start the teleop node with the c ommands:
  - *ros2 run turtlesim turtle_teleop_key*

Recording and playing back data

- Make a new directory to store our saved recordings:
  - *mkdir bag_files*
  - *cd bag_files*

- Record all topics:
  - *ros2 bag record --all*

- See details about recording by running:
  - *ros2 bag info <bag_file_name>*

```
$ ros2 bag info subset
Files:             subset.db3
Bag size:          228.5 KiB
Storage id:        sqlite3
Duration:          48.47s
Start:             Oct 11 2019 06:09:09.12 (1570799349.12)
End                Oct 11 2019 06:09:57.60 (1570799397.60)
Messages:          3013
Topic information: Topic: /turtle1/cmd_vel | Type: geometry_msgs/msg/Twist | Count: 9 | Serialization Format: cdr
                   Topic: /turtle1/pose | Type: turtlesim/msg/Pose | Count: 3004 | Serialization Format: cdr
```

Recording and playing back data

- Choose a topic: ros2 bag can only record data from published messages in topics

```
$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

Want to only record this topic

- ros2 bag record <topic_name>
  - *ros2 bag record /turtle1/cmd_vel*

Recording and playing back data

- Choose a topic: ros2 bag can only record data from published messages in topics

```
$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

Want to record multiple topics

- ros2 bag record <topic_name1> <topic_name2> …
  - *ros2 bag record /turtle1/cmd_vel /turtle1/pose*

Recording and playing back data

- Replay the data recorded in the bag file
  - *ros2 bag play <bagfile_name>*

Check it using rqt_plot