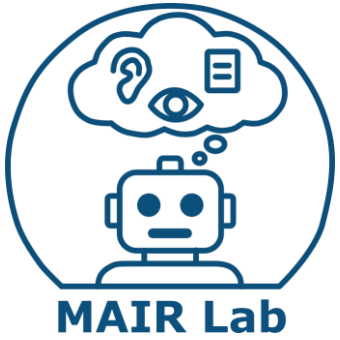
ROS2: Params, Actions

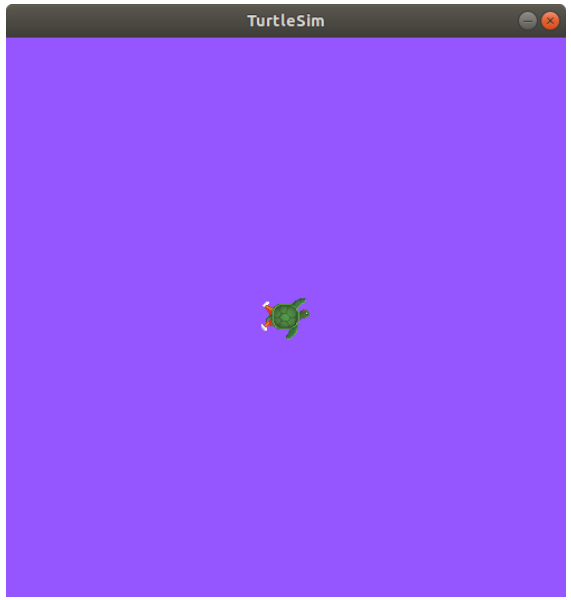
운영체제의 실제

안인규 (Inkyu An)



What is Parameters?

- A parameter is a configuration value of a node
- We can think of parameters as node settings
- A node can store parameters as integers, floats, Booleans, string, and lists: Each node maintains its own parameters



```
$ ros2 param list
/teleop_turtle:
  qos_overrides./parameter_events.publisher.depth
  qos_overrides./parameter_events.publisher.durability
  qos_overrides./parameter_events.publisher.history
  qos_overrides./parameter_events.publisher.reliability
  scale_angular
  scale_linear
  use_sim_time
/turtlesim:
  background_b
  background_g
  background_r
  qos_overrides./parameter_events.publisher.depth
  qos_overrides./parameter_events.publisher.durability
  qos_overrides./parameter_events.publisher.history
  qos_overrides./parameter_events.publisher.reliability
  use_sim_time
```

Prerequisites: 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 Parameters?

- *ros2 param list*
  - To see the parameters belonging to your nodes, open a new terminal and enter the command:

```
$ ros2 param list
/teleop_turtle:
  qos_overrides./parameter_events.publisher.depth
  qos_overrides./parameter_events.publisher.durability
  qos_overrides./parameter_events.publisher.history
  qos_overrides./parameter_events.publisher.reliability
  scale_angular
  scale_linear
  use_sim_time
/turtlesim:
  background_b
  background_g
  background_r
  qos_overrides./parameter_events.publisher.depth
  qos_overrides./parameter_events.publisher.durability
  qos_overrides./parameter_events.publisher.history
  qos_overrides./parameter_events.publisher.reliability
  use_sim_time
```

Determine the background color of the turtlesim window

What is Parameters?

- *ros2 param get <node_name> <parameter_name>*
  - To display the type and current value of a parameter, use the command:

```
$ ros2 param get /turtlesim background_g
Integer value is: 86
```

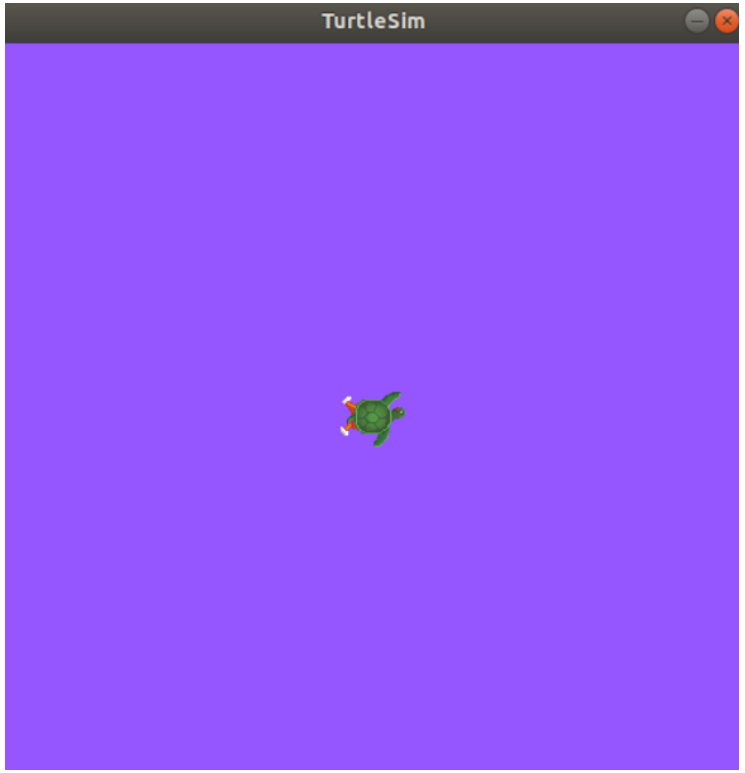
- Type: Interger
- Value: 86

→ Run the same command on other parameters of red and blue!

What is Parameters?

- *ros2 param set <node_name> <parameter_name> <value>*
  - To change a parameter's value at runtime. use the command:

```
$ ros2 param set /turtlesim background_r 150
Set parameter successful
```



Q: if I run the turtlesim node again, will the current settings be preserved?

What is Parameters?

- *ros2 param dump <node_name>*
  - We can view all of a node's current parameter values by using the comments:
    - *ros2 param dump /turtlesim*

  - The command prints to the standard output (stdout) by default
  - We can redirect the parameter values into a file

```
$ ros2 param dump /turtlesim > turtlesim.yaml
```

Q: What is it?

What is Parameters?

- *ros2 param dump <node_name>*
  - We can view all of a node's current parameter values by using the comments:
    - *ros2 param dump /turtlesim*

  - The command prints to the standard output (stdout) by default
  - We can redirect the parameter values into a file

```
$ ros2 param dump /turtlesim > turtlesim.yaml
```

Q: What is it?

```
/turtlesim:
  ros__parameters:
    background_b: 255
    background_g: 86
    background_r: 150
    qos_overrides:
      /parameter_events:
        publisher:
          depth: 1000
          durability: volatile
          history: keep_last
          reliability: reliable
    use_sim_time: false
```

What is Parameters?

- *ros2 param load <node_name> <parameter_file>*
  - We can load parameters from a file:
    - *ros2 param load /turtlesim turtlesim.yaml*

```
$ ros2 param load /turtlesim turtlesim.yaml
Set parameter background_b successful
Set parameter background_g successful
Set parameter background_r successful
Set parameter qos_overrides./parameter_events.publisher.depth failed: parameter 'qos_overrides./parameter_events.publisher.depth' cannot
Set parameter qos_overrides./parameter_events.publisher.durability failed: parameter 'qos_overrides./parameter_events.publisher.durabili
Set parameter qos_overrides./parameter_events.publisher.history failed: parameter 'qos_overrides./parameter_events.publisher.history' ca
Set parameter qos_overrides./parameter_events.publisher.reliability failed: parameter 'qos_overrides./parameter_events.publisher.reliabi
Set parameter use_sim_time successful
```

"qos_overrides" are read-only parameters
(They cannot be updated during runtime!)

"qos_overrides": parameters about QoS
(Quality of Service)

Why ROS2?

	ROS1		ROS2
구분	**TCPROS** (ROS1)	**UDPROS** (ROS1)	**DDS** (ROS2)
기반	TCP	UDP	산업 표준 DDS
신뢰성	높음 (순서/전송 보장)	낮음 (손실 가능)	**QoS(Quality of Service)에 따라 조정 가능**
지연 시간	상대적으로 큼	낮음 (빠름)	실시간 제어 가능
보안	없음	없음	암호화·인증 지원
확장성	제한적	제한적	멀티로봇/분산 환경 최적화
주 용도	일반 메시지 전송	대용량 센서 데이터	모든 로봇 통신 (범용)

DDS?

항목	**Fast DDS (eProsima)**	**Cyclone DDS (Eclipse)**	**Connext DDS (RTI)**	**GurumDDS (GurumNetworks)**
라이선스	Apache 2.0 (오픈소스)	EPL v2.0 (오픈소스)	Commercial / 연구용 무료	Commercial
기본 포함 여부	O (ROS 2 기본 포함 (Default))	O (ROS 2 기본 포함)	X (별도 설치 필요)	X (별도 설치 필요)
성능	빠른 discovery, 낮은 latency	안정적인 통신, 중간 정도 성능	매우 우수 (산업용 실시간 환경에 적합)	RTOS에 적합, 실시간 대응 강점
지원 플랫폼	Linux, macOS, Windows	Linux 중심	다수 OS (RTOS 포함)	RTOS 포함
메모리 사용량	중간	낮음 (임베디드에 적합)	높음 (기능이 많아서)	낮음
사용 복잡도	낮음 (설정 간단)	낮음	중간 이상 (라이선스 등록, 설정 필요)	중간
QoS 지원	대부분 지원	대부분 지원	완전 지원	대부분 지원

QoS (Quality of Service): ROS2의 통신의 "Reliability (신뢰성)", "Durability (지속성)", "Timing (타이밍)" 등을 설정하는 장치

What is Parameters?

- What is QoS?

```
$ ros2 param dump /turtlesim > turtlesim.yaml
```

```
/turtlesim:
  ros__parameters:
    background_b: 255
    background_g: 86
    background_r: 150
    qos_overrides:
      /parameter_events:
        publisher:
          depth: 1000
          durability: volatile
          history: keep_last
          reliability: reliable
    use_sim_time: false
```

- **depth**: The num. of messages to buffer
- **durability**: volatile vs. transient_local
  - volatile: 휘발성 (이전 message는 수신 불가능), e.g., sensor data
  - transient_local: 비휘발성 (버퍼에 있는 message 수신 가능)
- **history**: keep_last vs. keep_all
  - keep_last: 최근 N개의 message만 버퍼링
  - keep_all: 모든 message 버퍼링
- **reliability**: reliable vs. best_effort
  - reliable: message 전달 보장 (지연/재전송 가능)
  - best_effort: 가능한 빠르게

What is Parameters?

- *ros2 param load <node_name> <parameter_file>*
  - We can load parameters from a file:
    - *ros2 param load /turtlesim turtlesim.yaml*

```
$ ros2 param load /turtlesim turtlesim.yaml
Set parameter background_b successful
Set parameter background_g successful
Set parameter background_r successful
Set parameter qos_overrides./parameter_events.publisher.depth failed: parameter 'qos_overrides./parameter_events.publisher.depth' cannot
Set parameter qos_overrides./parameter_events.publisher.durability failed: parameter 'qos_overrides./parameter_events.publisher.durabili
Set parameter qos_overrides./parameter_events.publisher.history failed: parameter 'qos_overrides./parameter_events.publisher.history' ca
Set parameter qos_overrides./parameter_events.publisher.reliability failed: parameter 'qos_overrides./parameter_events.publisher.reliabi
Set parameter use_sim_time successful
```

"qos_overrides" are read-only parameters
(They cannot be updated during runtime!)

Q: How can we update our "qos_overrides" parameters?

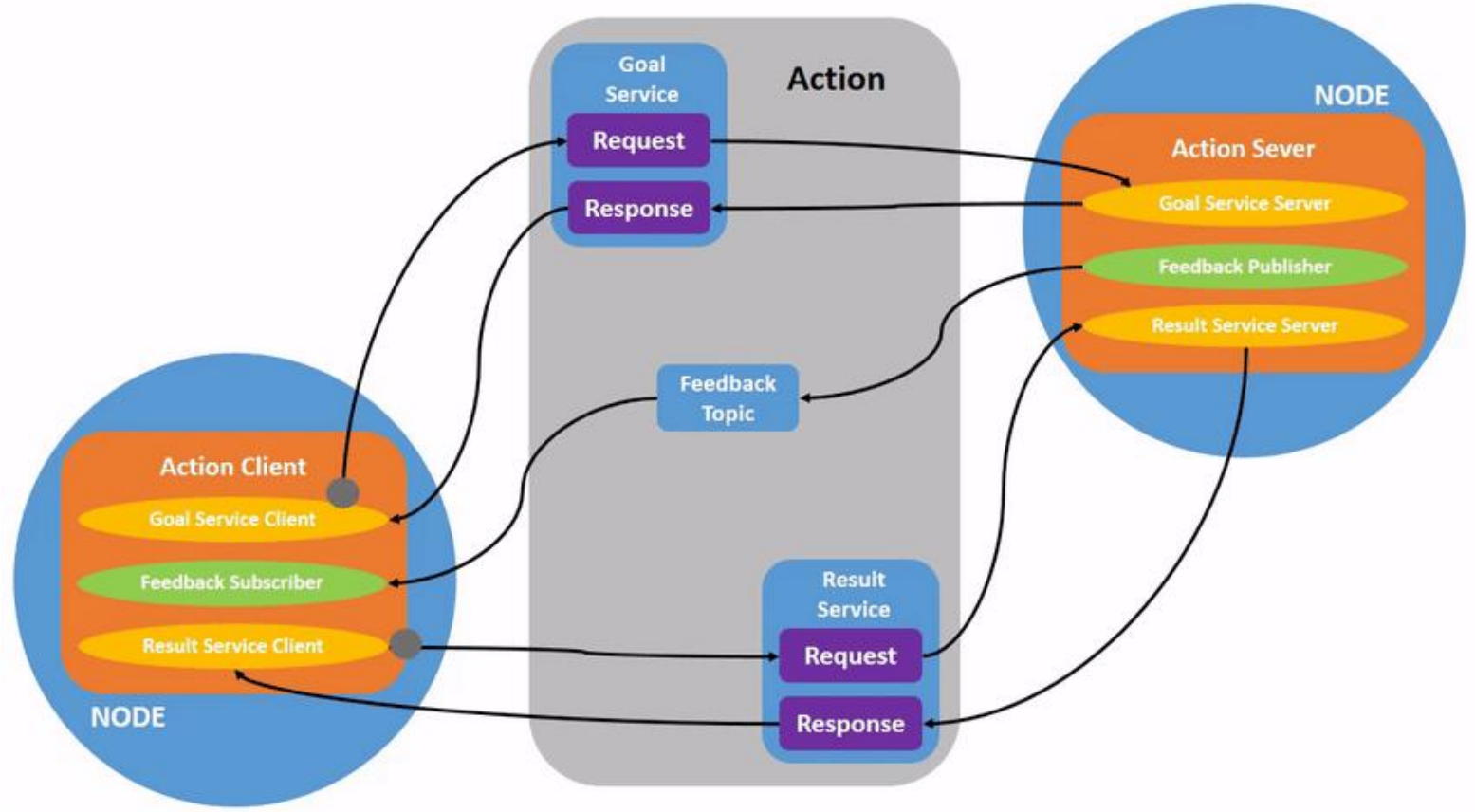
What is Parameters?

- Load parameter file on node startup
  - To start the same node using your saved parameter values:
    - *ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>*
  - Example:

```
$ ros2 run turtlesim turtlesim_node --ros-args --params-file turtlesim.yaml
```

What is Actions?

- Actions are one of the communication types in ROS2
- Actions are intended for long running tasks
- Actions consist of three parts: a goal, feedback, and a result



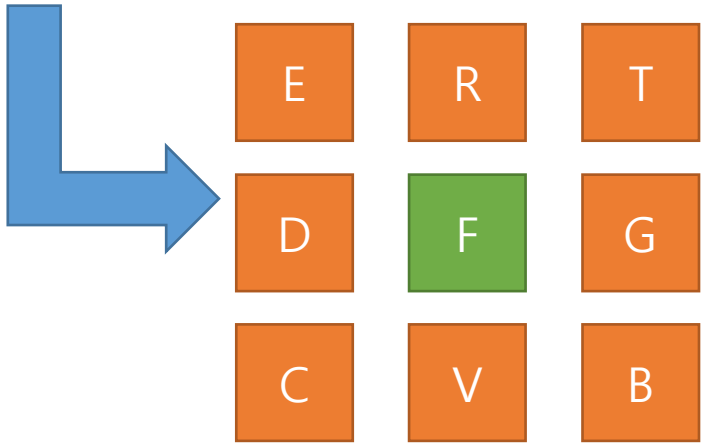
What is Actions?

- Actions are built on topics and services (The functionality is similar to services, except actions can be canceled)
- Actions also provide steady feedback, as opposed to services which return a single response
- Actions use a client-server model, similar to the publisher-subscriber model in Topic
  - An action client sends a goal to an action server that acknowledges the goal and returns a stream of feedback and a result

What is Actions?

- Use actions
  - When you launch the “/teleop_turtle” node, we will see the following message in our terminal:

```
Use arrow keys to move the turtle.
Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rotation.
```



1. [INFO] [turtlesim]: Rotation goal completed successfully
2. [INFO] [turtlesim]: Rotation goal canceled
3. [WARN] [turtlesim]: Rotation goal received before a previous goal finished. Aborting previous goal

What is Actions?

- Check *ros2 node info <node_name>*
  - To see the list of actions a node provides, do "*ros2 node /turtlesim*" and "*ros2 node /teleop_turtle*"

```
$ ros2 node info /turtlesim
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:
```

```
$ ros2 node info /teleop_turtle
/teleop_turtle
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Service Servers:
    /teleop_turtle/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /teleop_turtle/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /teleop_turtle/get_parameters: rcl_interfaces/srv/GetParameters
    /teleop_turtle/list_parameters: rcl_interfaces/srv/ListParameters
    /teleop_turtle/set_parameters: rcl_interfaces/srv/SetParameters
    /teleop_turtle/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:

  Action Clients:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
```

What is Actions?

- Check *ros2 node info <node_name>*
  - To see the list of actions a node provides, do "*ros2 node /turtlesim*" and "*ros2 node /teleop_turtle*"

```
$ ros2 node info /turtlesim
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:
```

Send goals

```
$ ros2 node info /teleop_turtle
/teleop_turtle
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Service Servers:
    /teleop_turtle/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /teleop_turtle/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /teleop_turtle/get_parameters: rcl_interfaces/srv/GetParameters
    /teleop_turtle/list_parameters: rcl_interfaces/srv/ListParameters
    /teleop_turtle/set_parameters: rcl_interfaces/srv/SetParameters
    /teleop_turtle/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:

  Action Clients:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
```

What is Actions?

- Check *ros2 node info <node_name>*
  - To see the list of actions a node provides, do "*ros2 node /turtlesim*" and "*ros2 node /teleop_turtle*"

```
$ ros2 node info /turtlesim
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:
```

Provides feedback

Sends goals

```
$ ros2 node info /teleop_turtle
/teleop_turtle
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Service Servers:
    /teleop_turtle/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /teleop_turtle/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /teleop_turtle/get_parameters: rcl_interfaces/srv/GetParameters
    /teleop_turtle/list_parameters: rcl_interfaces/srv/ListParameters
    /teleop_turtle/set_parameters: rcl_interfaces/srv/SetParameters
    /teleop_turtle/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:

  Action Clients:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
```

What is Actions?

- *ros2 action list*
  - To identify all the actions in the ROS graph, run the command:

```
$ ros2 action list
/turtle1/rotate_absolute
```

  - To identify types of all actions:

```
$ ros2 action list -t
/turtle1/rotate_absolute [turtlesim/action/RotateAbsolute]
```

What is Actions?

- *ros2 action info <action_name>*
  - We can further introspect the "/turtle1/rotate_absolute" action with the command:

```
$ ros2 action info /turtle1/rotate_absolute
Action: /turtle1/rotate_absolute
Action clients: 1
    /teleop_turtle
Action servers: 1
    /turtlesim
```

- *ros2 interface show*
  - To check the type of the action:

```
$ ros2 action list -t
/turtle1/rotate_absolute [turtlesim/action/RotateAbsolute]
```

type

```
$ ros2 interface show turtlesim/action/RotateAbsolute
```

What is Actions?

- *ros2 action info <action_name>*
  - We can further introspect the "/turtle1/rotate_absolute" action with the command:

```
$ ros2 action info /turtle1/rotate_absolute
Action: /turtle1/rotate_absolute
Action clients: 1
    /teleop_turtle
Action servers: 1
    /turtlesim
```

- *ros2 interface show*
  - To check the type of the action:

```
$ ros2 interface show turtlesim/action/RotateAbsolute
```

```
# The desired heading in radians
float32 theta
---
# The angular displacement in radians to the starting position
float32 delta
---
# The remaining rotation in radians
float32 remaining
```

Goal

Result

Feedback

What is Actions?

- *ros2 action send_goal <action_name> <action_type> <values>*
  - <values> needs to be in YAML format
  - *ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}"*

```
$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}"
Waiting for an action server to become available...
Sending goal:
   theta: 1.57

Goal accepted with ID: f8db8f44410849eaa93d3feb747dd444

Result:
  delta: -1.568000316619873

Goal finished with status: SUCCEEDED
```

What is Actions?

- *ros2 action send_goal <action_name> <action_type> <values>*
  - <values> needs to be in YAML format
  - To see the feedback of this goal, add "--feedback"
  - *ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: -1.57}" --feedback*

```
$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: -1.57}" --feedback
Sending goal:
   theta: -1.57

Goal accepted with ID: e6092c831f994afda92f0086f220da27

Feedback:
  remaining: -3.1268222332000732

Feedback:
  remaining: -3.1108222007751465

…

Result:
  delta: 3.1200008392333984

Goal finished with status: SUCCEEDED
```