ROS2_day3_HW1

20기 인턴 강은구

목차

5

1 topic

1.1 배경

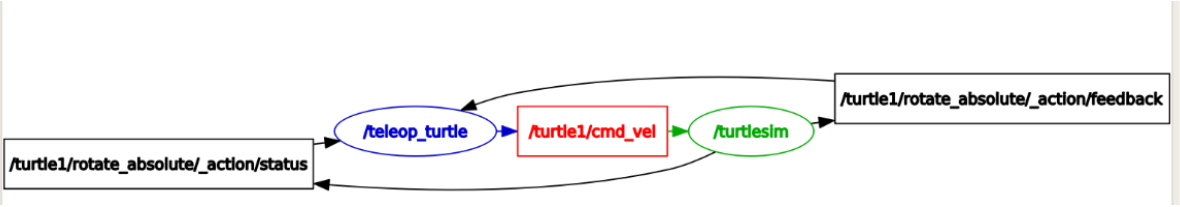
- Ros2는 복잡한 시스템을 모듈 형식의 노드로 나누어 관리한다. 각 노드는 메시지를 교환하며 원하는 정보를 송신 또는 수신한다.
- 노드는 모든 토픽에 데이터를 발행할 수 있으며, 동시에 모든 토픽을 구독하여 데이터를 얻을 수 있다.
- 토픽은 노드 간 데이터가 이동하는 주요 방법 중 하나이다.

1.2 적용-turltesim

- 터틀심이란?

  :터틀심은 통신 과정을 확인할 수 있는 대표적인 예제로, 여러 명령어를 통해 통신 과정에서 발생하는 동작이나 데이터 등을 확인할 수 있다.

  터틀심의 노드 간 연결 구조는 다음과 같다



  Teleop_turtle노드는 사용자가 입력한 키의 데이터를 turtlr1/cmd_vel토픽으로 발행하고, turtlesim노드는 이를 구독하여 발행된 데이터를 얻는다.

  이는 rqt_graph 명령어를 통해 확인할 수 있다.

- 전달되는 메시지의 형태

  Echo<topic_name> 명령어를 통해 teleop_turtle로붙터 turtle1/cmd_vel토픽을 통해 /turtlesim노드로 방행되는 정보를 확인할 수 있다. 키보드 입력이 있기 전까지 대기하다가 키보드 정보가 입력되면 teleop turtle이 입력된 정보를 발행함과 동시에 발행된 정보가 사용자에게 표시된다. 우리는 발행되는 정보가 다음과 같다는 것을 알 수 있다.

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

- 노드 간 연결성

토픽은 단순히 노드에서 노드 간의 연결이 아닌 여러 노드에서 한 노드로, 또는 하나의 노드에서 여러 노드로 동시에 발행이 가능하다. 결국 동일한 토픽을 등록한 발행자들은 모두 해당 토픽으로 발행할 수 있고, 발행된 정보는 해당 토픽이 등록된 모든 구독자들에게 전달된다.

- 연결 조건-메시지 형태

  발행자 노드와 구독자 노드 사이에 메시지가 전달되기 위해서는 양 쪽에서 지정한 메시지의 형태가 동일해야 한다. Turtlesim의 경우, geometry_msgs패키지의 Twist형태의 메시지로 서로 통신하기 때문에, 발행자 측에서 다른 형태의 메시지를 보내거나, 구독자 측에서 다른 형태의 메시지 형태를 요구하면, 정상적인 통신이 이루어지지 않는다.

  turtlesim에서 정상적인 정보 전달을 위해서는 이전에 확인한 메시지 형태를 통해 leaner.x~z, angular.x~z에 대한 데이터를 지정된 변수에 저장하여 발행해야만 구독자 노드에서 제대로된 정보를 인식하고 거북이를 움지일 수 있다.

  사용자가 직접 정보를 전달하기 위해서는 다음과 같은 명령어를 사용해야 한다.

```
$ ros2 topic pub /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}}"
```

- Hz

  데이터가 전송되는 데 있어서, 전송 속도가 존재한다. Hz /turtle1/pose라는 명령어를 통해 확인한 turtlesim의 전송속도는 다음과 같다.

```
$ ros2 topic hz /turtle1/pose
average rate: 59.354
  min: 0.005s max: 0.027s std dev: 0.00284s window: 58
```

  해당 속도는 발행 속도를 1hz설정했을 경우이다.

- 토픽 찾기

  해당 유형의 사용 가능한 토픽은 "ros2 topic finc<topc_type>을 통해 확인할 수 있다. Turtlesim의 경우 /turtle1/cmd_vel토픽을 확인할 수 있을 것이다. 즉, 키보드로 입력을 받아 거북이의 동작에 반영되기까지 데이터는 turtle1/cmd_vel토픽을 통해 전달된다는 것을 알 수 있다.

2 서비스(Service

2.1 배경

서비스는 ros그래프에서 노드 간의 또 다른 통신 방식이다. 토픽과 차이점은, 토픽의 경우 지속적으로 데이터를 발행할 수 있지만, 서비스의 클라이언트가 구체적으로 요청할 시에만 데이터를 제공한다.

2.2 서비스 목록

터틀심의 경우 다음과 같은 서비스 목록을 확인할 수 있다.

```
$ ros2 service list
/clear
/kill
/reset
/spawn
/teleop_turtle/describe_parameters
/teleop_turtle/get_parameter_types
/teleop_turtle/get_parameters
/teleop_turtle/list_parameters
/teleop_turtle/set_parameters
/teleop_turtle/set_parameters_atomically
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
```

여기서, teleop_turtle과 turtle1노드는 paramerters라는 동일한 여섯 개의 서비스가 포함된 것을 알 수 있다. 대부분의 ros2의 노드에는 매개변수가 굿축되는 이런 인프가 서비스가 있다.

2.3 서비스 유형

서비스에는 요청 및 응답 데이터의 구조를 설명하는 유형이 존재한다. 토픽과 유사하지만 서비스 유형의 경우 두 부분으로 구성된다.

turtlesim의 경우 사용되는 Empty유형은 서비스 호출이 요청을 할 때 데이터를 보내지 않고, 응답을 받을 때 데이터를 받지 않는다는 것을 의미한다.

2.4 Ros2 서비스 목록

ros2의 서비스 목록은 "ros2 service list –t" 명령어를 통해 확인할 수 있다.

```
$ ros2 service list
/clear
/kill
/reset
/spawn
/teleop_turtle/describe_parameters
/teleop_turtle/get_parameter_types
/teleop_turtle/get_parameters
/teleop_turtle/list_parameters
/teleop_turtle/set_parameters
/teleop_turtle/set_parameters_atomically
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
```

turtlesim이 사용하는 서비스로 clear, kill(원하는 거북이를 죽이는 서비스), reset(처음 상태로 되돌리는 서비스), set_pen(거북이가 그리는 궤적의 색, 크기를 지정하는 서비스) 등을 확인할 수 있다.
해당 사용자는 명령어를 통해 원하는 서비스의 동작을 요청할 수 있다.

2.5 서비스 찾기

사용자는 "ros2 service find<type_name>명령어를 통해 원하는 유형의 서비스를 찾을 수 있다. Empty의 경우, /clear, /reset서비스를 확인할 수 있다.

2.6 요청 및 응답 유형

"ros2 interface show turtlesim/srv/Sqawn"명령어를 통해 다음과 같이 터틀심의 응답 및 요청 유형을 확인할 수 있다.

```
$ ros2 interface show turtlesim/srv/Spawn
float32 x
float32 y
float32 theta
string name # Optional.  A unique name will be created and returned if this is empty
---
string name
```

해당 화면을 통해 호출하는데 필요한 인수로 /spawn/x/y/theta/name가 있음을 확인할 수 있다.

2.7 서비스 호출

위에서 확인한 서비스 유형을 이용해 직접 서비스를 요청할 수 있다.

- Ros2 service call <service_name> <service_type><arguments>
- Ros2 service call /clear std_srvs/srv/Empty: 거북이가 그니 선이 창에서 지워진다.
- Ros2 service call /spawn turtlesim/srve /Spawn"{x:?, y=?, theta:?, name='?'}:
  이와 같은 형식을 사용하여 서비스를 요청할 수 있다.

```
$ ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: ''}"
requester: making request: turtlesim.srv.Spawn_Request(x=2.0, y=2.0, theta=0.2, name='')

response:
turtlesim.srv.Spawn_Response(name='turtle2')
```

명령어를 입력하면 볼 수 있는 위 화면에서, 우리는 요청과 응답 내용을 확인할 수 있다. 요청자가 거북이의 위치 및 각도, 이름 정보를 전달하면, 응답자는 해당 요청을 처리한 다음에 name이라는 인수로 응답한다.

3 매개변수

3.1 배경

매개변수는 노드를 구성하는 값들이다. 노드는 매개변수의 값을 정수, 소수 등

의 값으로 저장할 수 있다.

3.2 매개변수 목록

turtlesim의 경우 "ros2 param list" 명령어를 통해 다음과 같이 매개변수 목록을 확인할 수 있다.

```
$ 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
```

turtlesim의 경우 매개변수 b,g,r을 통해 배경 색을 조절한다는 것을 알 수 있다. 예를 들어, "ros2 param set /turtlesim background_g <value>"명령어를 통해 g에 해당하는 값을 지정해주면 해당 값이 반영된 색깔로 배경 색이 변경되는 것을 볼 수 있다.

3.3 매개변수 값 확인

"ros2 param dump<node_name>명령어를 통해 다음과 같이 저장된 매개변수를 확인할 수 있다.

```
/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
```

위에서 다뤘던 g값 또한 사진과 같이 255라는 값이 저장되어있음을 알 수 있다.

3.4 정리

turtlesim에서 확인할 수 있듯이 배경색과 같은 요소는 각 노드에서 관리하게된다. 이때, 동작을 지정할 값은 모두 매개변수 값에 저장되며, rgb의 경우, 각각의 매개변수에 저장된 정수값을 통해 터틀심의 배경 색을 구성하게 된다. 즉, 매개변수는 노드를 구성하는 값이라고 이해할 수 있다.