

hw2 보고서

20기 인턴 송수민

1) 실행 결과

- py와 py

```
ssm@miki: ~/colcon_ws/day1/hw2
[INFO] [1757659375.450823460] [helloworld_publisher]: Published message: Hello W
orld: 14
[INFO] [1757659376.451552966] [helloworld_publisher]: Published message: Hello W
orld: 15
[INFO] [1757659377.451419936] [helloworld_publisher]: Published message: Hello W
orld: 16
[INFO] [1757659378.450195215] [helloworld_publisher]: Published message: Hello W
orld: 17
[INFO] [1757659379.451153373] [helloworld_publisher]: Published message: Hello W
orld: 18
[INFO] [1757659380.450185851] [helloworld_publisher]: Published message: Hello W
orld: 19
[INFO] [1757659381.451011967] [helloworld_publisher]: Published message: Hello W
orld: 20
[INFO] [1757659382.450906760] [helloworld_publisher]: Published message: Hello W
orld: 21
[INFO] [1757659383.451206684] [helloworld_publisher]: Published message: Hello W
orld: 22
[INFO] [1757659384.450964722] [helloworld_publisher]: Published message: Hello W
orld: 23
[INFO] [1757659385.450803691] [helloworld_publisher]: Published message: Hello W
orld: 24
```

- c++과 c++

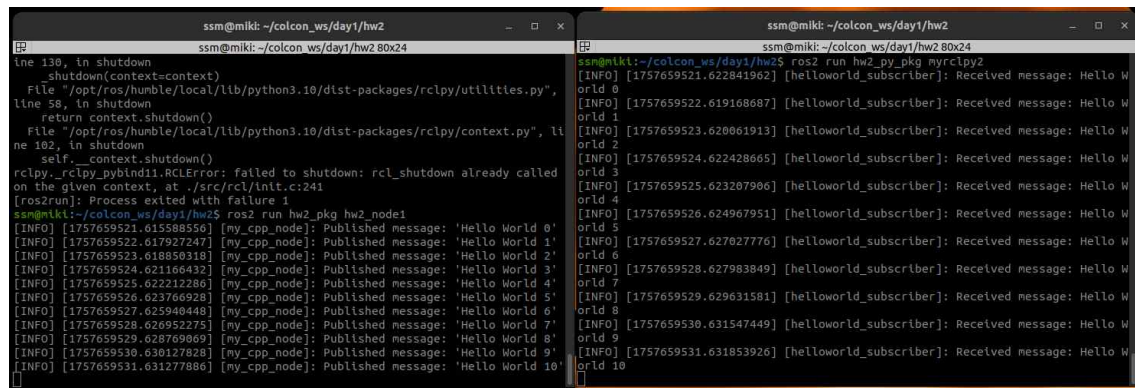
```
ssm@miki: ~/colcon_ws/day1/hw2
[INFO] [1757659446.953290236] [my_cpp_node]: Published message: 'Hello World 0'
[INFO] [1757659447.953384492] [my_cpp_node]: Published message: 'Hello World 1'
[INFO] [1757659448.953380490] [my_cpp_node]: Published message: 'Hello World 2'
[INFO] [1757659449.953358628] [my_cpp_node]: Published message: 'Hello World 3'

ssm@miki: ~/colcon_ws/day1/hw2
[INFO] [1757659447.953384492] [my_cpp_node]: Published message: 'Hello World 0'
[INFO] [1757659448.953380490] [my_cpp_node]: Published message: 'Hello World 1'
[INFO] [1757659449.953358628] [my_cpp_node]: Published message: 'Hello World 2'
[INFO] [1757659450.953358628] [my_cpp_node]: Published message: 'Hello World 3'
```

- py 와 c++

```
ssm@miki: ~/colcon_ws/day1/hw2
[INFO] [1757659487.240535679] [helloworld_publisher]: Published message: Hello W
orld: 1
[INFO] [1757659488.240397050] [helloworld_publisher]: Published message: Hello W
orld: 2
[INFO] [1757659489.240852865] [helloworld_publisher]: Published message: Hello W
orld: 3
[INFO] [1757659490.240571118] [helloworld_publisher]: Published message: Hello W
orld: 4
[INFO] [1757659491.240871554] [helloworld_publisher]: Published message: Hello W
orld: 5
[INFO] [1757659492.240806114] [helloworld_publisher]: Published message: Hello W
orld: 6
[INFO] [1757659493.240729336] [helloworld_publisher]: Published message: Hello W
orld: 7
[INFO] [1757659494.240941623] [helloworld_publisher]: Published message: Hello W
orld: 8
[INFO] [1757659495.240729624] [helloworld_publisher]: Published message: Hello W
orld: 9
[INFO] [1757659496.240246779] [helloworld_publisher]: Published message: Hello W
orld: 10
```

- c++과 py



```
ssm@miki: ~/colcon_ws/day1/hw2
line 138, in shutdown
    shutdown(context=context)
File ~/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/utilities.py,
line 58, in shutdown
    return context.shutdown()
File ~/opt/ros/humble/local/lib/python3.10/dist-packages/rclpy/context.py, li
ne 102, in shutdown
    self._context.shutdown()
rclpy._rclpy_pybind11.RclError: failed to shutdown: rcl_shutdown already called
on the given context, at ./src/rcl/init.c:241
[ros2run]: Process exited with failure 1
ssm@miki:~/colcon_ws/day1/hw2$ ros2 run hw2_pkg hw2_node1
[INFO] [1757659521.615588556] [my_cpp_node]: Published message: 'Hello World 0'
[INFO] [1757659522.617927247] [my_cpp_node]: Published message: 'Hello World 1'
[INFO] [1757659523.618850318] [my_cpp_node]: Published message: 'Hello World 2'
[INFO] [1757659524.621160432] [my_cpp_node]: Published message: 'Hello World 3'
[INFO] [1757659525.622212286] [my_cpp_node]: Published message: 'Hello World 4'
[INFO] [1757659526.623766928] [my_cpp_node]: Published message: 'Hello World 5'
[INFO] [1757659527.625940448] [my_cpp_node]: Published message: 'Hello World 6'
[INFO] [1757659528.626952275] [my_cpp_node]: Published message: 'Hello World 7'
[INFO] [1757659529.628769069] [my_cpp_node]: Published message: 'Hello World 8'
[INFO] [1757659530.630127828] [my_cpp_node]: Published message: 'Hello World 9'
[INFO] [1757659531.631277886] [my_cpp_node]: Published message: 'Hello World 10'

ssm@miki:~/colcon_ws/day1/hw2$ ros2 run hw2_py_pkg myrclpy2
[INFO] [1757659521.622841962] [helloworld_subscriber]: Received message: Hello W
orld 0
[INFO] [1757659522.619168687] [helloworld_subscriber]: Received message: Hello W
orld 1
[INFO] [1757659523.620061913] [helloworld_subscriber]: Received message: Hello W
orld 2
[INFO] [1757659524.622428665] [helloworld_subscriber]: Received message: Hello W
orld 3
[INFO] [1757659525.623287906] [helloworld_subscriber]: Received message: Hello W
orld 4
[INFO] [1757659526.624967951] [helloworld_subscriber]: Received message: Hello W
orld 5
[INFO] [1757659527.627027776] [helloworld_subscriber]: Received message: Hello W
orld 6
[INFO] [1757659528.627983849] [helloworld_subscriber]: Received message: Hello W
orld 7
[INFO] [1757659529.629631581] [helloworld_subscriber]: Received message: Hello W
orld 8
[INFO] [1757659530.631547449] [helloworld_subscriber]: Received message: Hello W
orld 9
[INFO] [1757659531.631853926] [helloworld_subscriber]: Received message: Hello W
orld 10
```

2) 코드 분석

1. c++ 패키지

1.1 헤더 파일

```
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

class MyCppNode : public rclcpp::Node
{
public:
    MyCppNode();

private:
    void timer_callback();
    rclcpp::Publisher<std_msgs::msg::String>::SharedPtr mycpp_publisher_;
    rclcpp::TimerBase::SharedPtr timer_;
    int count_ = 0;
};

class MyCppSubscriber : public rclcpp::Node
{
public:
    MyCppSubscriber();

private:
    void topic_callback(const std_msgs::msg::String::SharedPtr msg);
    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr mycpp_subscriber_;
};
```

통신을 하기위해 예제를 참고하여 송신자와 수신자 클래스를 작성함

1.2 송신자 파일

- 공통 부분

```
#include "hw2_pkg/my_cpp_node.hpp"
#include <chrono>
using namespace std::chrono_literals;

MyCppNode::MyCppNode() : Node("my_cpp_node")
{
    mycpp_publisher_ = this->create_publisher<std_msgs::msg::String>("helloworld", 10);
    timer_ = this->create_wall_timer(
        1s, std::bind(&MyCppNode::timer_callback, this));
}

void MyCppNode::timer_callback()
{
    auto msg = std_msgs::msg::String();
    msg.data = "Hello World " + std::to_string(count_++);
    RCLCPP_INFO(this->get_logger(), "Published message: '%s'", msg.data.c_str());
    mycpp_publisher_->publish(msg);
}

MyCppSubscriber::MyCppSubscriber() : Node("my_cpp_subscriber")
{
    mycpp_subscriber_ = this->create_subscription<std_msgs::msg::String>(
        "helloworld", 10,
        std::bind(&MyCppSubscriber::topic_callback, this, std::placeholders::_1));
}

void MyCppSubscriber::topic_callback(const std_msgs::msg::String::SharedPtr msg)
{
    RCLCPP_INFO(this->get_logger(), "Received message: '%s'", msg->data.c_str());
}
```

MyCppNode 생성자. Node("my_cpp_node")로 부모 클래스를 초기화하며 노드 이름을 지정한다. create_publisher로 "helloworld" 토픽 송신자를 만들고, 1초마다 timer_callback()이 호출되도록 create_wall_timer를 설정한다.

timer_callback 함수. std_msgs::msg::String 메시지에 "Hello World "와 증가하는 숫자를 붙여 전송한다. RCLCPP_INFO로 터미널에 로그 출력 후, 송신자를 통해 메시지 발행한다.

MyCppSubscriber 생성자. Node("my_cpp_subscriber")로 초기화하며 노드 이름을 지정한다. create_subscription으로 "helloworld" 토픽 구독자를 만들고, 메시지를 수신하면 topic_callback()이 호출되도록 연결한다.

topic_callback 함수. 구독한 메시지를 수신하면 RCLCPP_INFO로 메시지 내용을 로그로 출력하여 확인 가능하게 한다.

- main

```
int main(int argc, char **argv)
{
    rclcpp::init(argc, argv);

    auto publisher_node = std::make_shared<MyCppNode>();

    rclcpp::executors::MultiThreadedExecutor executor;
    executor.add_node(publisher_node);
    executor.spin();

    rclcpp::shutdown();
    return 0;
}
```

ROS 2를 초기화하고, 송신자 노드를 만든다. 멀티스레드 실행기에 등록하고 spin을 시작한다. spin이 도는 동안 송신자가 메시지를 계속 보내고 콜백이 처리된다. 종료하면 rclcpp를 정리하고 프로그램을 끝낸다.

1.3 수신자 파일

- main

```
int main(int argc, char **argv)
{
    rclcpp::init(argc, argv);

    auto subscriber_node = std::make_shared<MyCppSubscriber>();

    rclcpp::executors::MultiThreadedExecutor executor;
    executor.add_node(subscriber_node);
    executor.spin();

    rclcpp::shutdown();
    return 0;
}
```

송신자와 달리 수신자 노드를 생성하고 멀티스레드 실행기에 등록하고 spin을 시작한다. spin이 도는 동안 수신자가 메시지를 계속 받고 콜백이 처리된다. 종료하면 rclcpp를 정리하고 프로그램을 끝낸다.

2. py 패키지

2.1 송신자 파일

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class HelloworldPublisher(Node):

    def __init__(self):
        super().__init__('helloworld_publisher')
        self.helloworld_publisher = self.create_publisher(String, 'helloworld', 10)
        self.timer = self.create_timer(1, self.publish_helloworld_msg)
        self.count = 0

    def publish_helloworld_msg(self):
        msg = String()
        msg.data = 'Hello World: {}'.format(self.count)
        self.helloworld_publisher.publish(msg)
        self.get_logger().info('Published message: {}'.format(msg.data))
        self.count += 1

def main(args=None):
    rclpy.init(args=args)
    node = HelloworldPublisher()
    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt (SIGINT)')
    finally:
        node.destroy_node()
        rclpy.shutdown()

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

MyCppNode 생성자

Node("my_cpp_node")로 부모 클래스를 초기화하며 노드 이름을 지정한다.

create_publisher로 "helloworld" 토픽 퍼블리셔를 만들고, 1초마다 timer_callback()이 호출되도록 create_wall_timer를 설정한다.

timer_callback 함수

std_msgs::msg::String 메시지에 "Hello World "와 증가하는 숫자를 붙여 전송한다.
RCLCPP_INFO로 터미널에 로그 출력 후, 퍼블리셔를 통해 메시지를 발행한다.

MyCppSubscriber 생성자

Node("my_cpp_subscriber")로 초기화하며 노드 이름을 지정한다.

create_subscription으로 "helloworld" 토픽 구독자를 만들고, 메시지를 수신하면 topic_callback()이 호출되도록 연결한다.

topic_callback 함수

구독한 메시지를 수신하면 메시지 내용을 로그로 출력하여 확인 가능하게 한다.

main 함수

ROS 2를 초기화하고 송신자 노드를 만든다.

멀티스레드 실행기에 등록하고 spin을 시작한다.

spin이 도는 동안 송신자가 메시지를 계속 보내고 콜백이 처리된다.

2.2 수신자 파일

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class HelloworldSubscriber(Node):

    def __init__(self):
        super().__init__('helloworld_subscriber')
        self.helloworld_subscriber = self.create_subscription(
            String,
            'helloworld',
            self.subscribe_topic_message,
            10)

    def subscribe_topic_message(self, msg):
        self.get_logger().info('Received message: {0}'.format(msg.data))

def main(args=None):
    rclpy.init(args=args)
    node = HelloworldSubscriber()
    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        node.get_logger().info('Keyboard Interrupt (SIGINT)')
    finally:
        node.destroy_node()
        rclpy.shutdown()

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

HelloworldSubscriber 생성자

Node("helloworld_subscriber")로 부모 클래스를 초기화하며 노드 이름을 지정한다.

create_subscription으로 "helloworld" 토픽 구독자를 만들고, 메시지를 수신하면 subscribe_topic_message()가 호출되도록 연결한다.

subscribe_topic_message 함수

구독한 메시지를 수신하면 메시지 내용을 출력하여 확인 가능하게 한다.

main 함수

ROS 2를 초기화하고 구독자 노드를 만든다.

노드를 계속 돌려서 콜백이 메시지를 처리하게 한다.