DAY2
과제 1번 보고서

2023741024
로봇학부 박건후

목차

(1) msg 파일에 대한 설명

위 사진은 제가 msg파일에서 선언한 부분입니다. ros에 내장되어 있는 int32를 벡터로 선언한 부분입니다. 이렇게 선언을 하면 이 msg파일을 사용하는 파일에게 해당 파일을 기반으로 한 hpp파일이 자동으로 만들어집니다. 그리고 이를 인클루드하면 선언한 것을 이름으로 타입을 사용할 수 있게 되고 그 타입의 멤버로 msg파일에 저장된 멤버들이 있게되는 것입니다. 이 파일을 사용하기 위해서는 CMakeList와 package.xml에 msg 패키지명을 추가해줘야 합니다.

이런 정보와 코드 예제를 인터넷 서칭을 통해 참고를 하여 만들 수 있었습니다. 아래의 링크는 제가 과제를 하며 참고를 하게 된 사이트입니다.

https://velog.io/@i_robo_u/%EA%B7%B8%EB%9E%98%EC%84%9C-%EC%96%B4%EB%96%BB%EA%B2%8C-%EB%8C%80%ED%99%94%ED%95%A0%EA%B1%B4%EB%8D%B0-ROS2-%EB%82%B4-msg%EC%99%80-srv-%EB%A7%8C%EB%93%A4%EA%B8%B0

```
#include "rclcpp/rclcpp.hpp"
#include "tutorial_interfaces/msg/num.hpp"

publisher_ = this->create_publisher<tutorial_interfaces::msg::Num>("topic", 10);
```

이렇게 인클루드하고, 이 인클루드한 타입을 사용하는 부분까지 학습할 수 있었습니다.

(2) 헤더 파일

```
#include <rclcpp/rclcpp.hpp>
#include "custom_interfaces/msg/colcon.hpp" // 사용자 정의한 메시지 타입 헤더 파일 포함

class publisher : public rclcpp::Node //퍼블리셔
{

    rclcpp::Publisher<custom_interfaces::msg::Colcon>::SharedPtr pub; //  인클루드한
   public:
    publisher();
    void publish_data();


};
class subscriber : public rclcpp::Node //서브스크라이버
{
    rclcpp::Subscription<custom_interfaces::msg::Colcon>::SharedPtr sub; //  인클루

   public:
    subscriber();
    void callback(const custom_interfaces::msg::Colcon::SharedPtr msg);//메시지를 받
};
```

해당 msg파일을 사용하기 위해 msg파일의 패키지명이 custom_interfaces이므로 그것과 msg를 붙이고 msg파일명을 추가하여 인클루드 선언을 하였습니다. 그리고 이를 통해 정보를 담아 전달해야 하기에 인클루드한 것의 타입을 써야 합니다. 그래서 publish를 담당하는 클래스에서는 custom_interfaces::msg::Colcon으로 인스턴스화하였습니다. 그리고 메시지를 받는 subscriber 클래스에서는 메시지를 받을 경우, callback()메서드를 콜백함수로 호출하도록 설계했고, 메시지에 대한 객체에 대한 SharedPtr을 사용하여 이를 가리키도록 만들었습니다.

(3) 구현 파일

```
#include "hw1.hpp"
#include <iostream>

publisher::publisher() : Node("publisher") //노드명 설정
{
    pub = this->create_publisher<custom_interfaces::msg::Colcon>("topic", 10); /
}
void publisher:: publish_data(){ //데이터 발행 함수
    int num=0;
    custom_interfaces::msg::Colcon data;
    RCLCPP_INFO(this->get_logger(), "10개의 정수를 입력하세요: "); //콘솔에 출력
    for(int i=0; i< 10; i++){
    std::cin>> num;
    if(!std::cin){ //예외 처리함
        std::cout<<"다시 입력하세요"<<std::endl;
        std::cin.clear();
        std::cin.ignore(1000, '\n');
        i--;    //잘못 입력받았으니 다시 입력받도록 입력받는 횟수 처리
        continue;
    }
```

publisher의 구현부를 보겠습니다. 생성자에서는 노드명을 설정하고 create헬퍼 함수를 통해 객체를 생성하고 포인터인 pub를 초기화했습니다.

publish_data()에서는 숫자 10개를 입력받고 이를 msg파일에서 만든 벡터에 넣고 한 번에 보내야 합니다. 그렇게 하기 위해 인클루드하여 얻은 custom_interfaces::msg::Colcon이라는 타입의 객체를 생성합니다(객체명 data). 이 객체에 10개의 정수를 넣어서 이 객체를 publish하여 보내면 되기 때문입니다. 해당 타입에는 data라는 int32[]벡터가 있습니다. 이것이 벡터라고 불리는 이유는 실제로 hpp파일로 헤더파일이 구성될 때 그 클래스의 멤버가 벡터로 구현되기 때문입니다. 그래서 custom_interfaces::msg::Colcon이라는 타입 안에 data라는 이름의 벡터 멤버가 있습니다. 그래서 이 벡터는 int32로 인스턴스화 되어 있을 것이기에 10개의 정수값을 넣으면 될 것입니다.

우선 10개의 값을 받기 위해 10개의 정수를 입력하라는 것을 터미널 창에 출력하게 하고 10개의 정수를 입력받았습니다. 그리고 if문을 통해 예외 처리를 진행했습니다.

```
    int integer = num;

    data.data.push_back(integer); // msg파일에서 []를 사용할 경우 이는 cpp파일로 만들어질 때 벡터로 변환되기
    //msg파일에서 data라고 이름을 지었기에 data라는 벡터 멤버가 있는 것
    }
    pub->publish(data);

}
void subscriber::callback(const custom_interfaces::msg::Colcon::SharedPtr msg){ // 메시지를 받았을

    RCLCPP_INFO(this->get_logger(), "받은 데이터: ");
    for(auto i : msg->data){

    RCLCPP_INFO(this->get_logger(), "%d ", (i)); //받은 모든 데이터를 잘 받았음을 확인하기 위해 각각 출력함
    }
}
subscriber::subscriber() : Node("subscriber") //노드명 설정
{
    sub = this->create_subscription<custom_interfaces::msg::Colcon>(
        "topic", 10, std::bind(&subscriber::callback, this, std::placeholders::_1)); //서브스크립
}
```

정수를 입력받으면 int에 저장하여 이를 push_back()을 하여 data라는 벡터에 넣었습니다. 이 때 객체명과 겹치게 코딩하다보니 data.data.pushback이라는 코드를 짜게 되었습니다. 이렇게 입력받고 벡터에 저장하는 것을 10번 반복하면 for문을 빠져나와 publish로 객체를 보내면 10개의 int값을 한 번에 넣는 효과를 낼 수 있습니다.

그 아래는 subscriber의 callback메서드 구현부입니다. 여기서는 메시지를 받고 해당 전달받은 객체를 SharedPtr로 가리켜서 그 정보를 가져올 수 있습니다. 받은 객체의 data 멤버가 벡터이므로 이 벡터에 접근하여 정보를 가져와야 합니다. 그래서 이번에 범위 기반 for문을 사용하여 벡터으 데이터를 하나씩 가져와서 그 정보를 잘 받았다는 것을 확인하기 위해 터미널 창에 각 숫자를 모두 띄우도록 코딩했습니다.

callback메서드 아래 부분은 subscriber의 생성자입니다. 이 부분은 하던대로 노드명 설정하고 토픽명 설정하며 콜백함수를 바인딩해주면 됩니다.

(4) publish main 파일

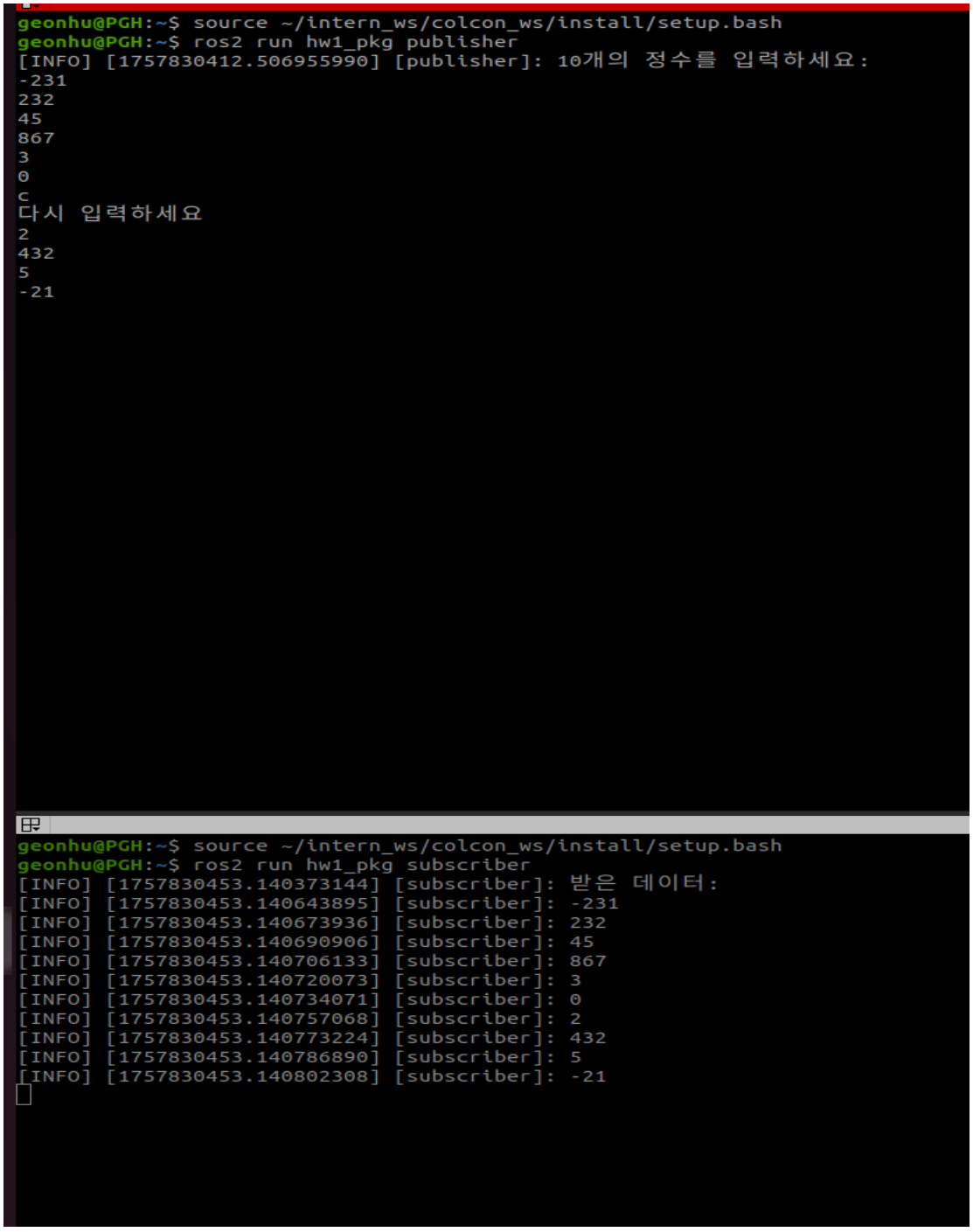
```
#include "hw1.hpp"
int main(int argc, char **argv)
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<publisher>();
    node->publish_data(); //spin만 하다보면 해당 publish_data가 호
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

마지막으로 publish의 main의 구현부입니다. 여기서 기존에 하던 방식에 추가로 publish_data라는 메서드를 여기서 출력해주었습니다. 그 이유는 spin을 하게 둘 경우 publish_data라는 메서드가 호출되지 않을 수 있기 때문입니다. 그래서 명시적으로 spin 이전에 node->publish_data를 호출하여 프로그램이 시작되면 publisher객체를 생성하고 바로 입력을 받을 수 있도록 한 것입니다.

```
#include "hw1.hpp"
int main(int argc, char **argv)
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<subscriber>();
    rclcpp::spin(node);
    rclcpp::shutdown();

    return 0;
}
```

subscriber의 경우 별 다르게 특별히 다르게 둔 부분이 없이 객체를 생성하고 메시지 올 때까지 대기하도록 했습니다.



위 사진은 실행 결과를 나타내는 사진입니다. 위 터미널에서는 값을 입력하여 10개를 다 입력하면 그 후에 subscriber에서 받았던 값을 모두 출력해주는 것입니다. 이 때 c라는 문자를 중간에 넣었는데 알맞게 예외처리를 하는 부분까지 넣었습니다. 감사합니다.