Ros 7w

C++ Style

1. C++14 Standard를 준수 한다
2. 라인길이 최대 100문자
3. 이름 규칙(Naming)

-. Camelcased : 타입 , 클래스, 구조체, 열거형

-. snake_case : 파일, 패키지, 구조체, 열거형

-. ALL_CAPITALS : 상수, 매크로

-. 소스 파일은 cpp 확장자 사용

-. 헤더 파일은 hpp 확장자 사용

-. 전역 변수(Global variable)를 반드시 사용해야 할 경우 접두어(g_)를 붙인다

-. 클래스 멤버 변수는 마지막에 밑줄(_)을 붙인다

4. 공백 문자 대 탭 (Spaces vs Tabs)

-. 기본들여 쓰기(Indent)는 공백문자(Space)2개를 사용한다(Tab 문자 사용 금지)

-. Class 의 접근 지정자 (public,protected,private)는 들여쓰기를 하지 않는다

5. 괄호(Brace)

-. if, else, do, while, for 구문에 괄호를 사용한다

6. 주석(Comments)

-. 문서 주석은/** */을 사용한다

-. 구현 주석은 //을 사용한다

7. 린터(Linters)

-. C++코드 스타일의 자동 오류 검출을 위하여 ament_cpplint , ament_uncrustify를 사용하고 정적 코드 분석이 필요한 경우 ament_cppcheck를 사용한다

8. 기타

-. Boost 라이브러리의 사용은 가능한 피하고 어쩔수 없을 경우에만 사용한다

-. 포인터 구문은 char * c처럼 사용한다 (char* c나 char *c 처럼 사용하지 않는다)

-. 중첩 템플릿은 set<list<srting>>처럼 사용한다(set<list<sring>> 또는 set< list<string> > 처럼 사용하지 않는다

Python Style

1. 파이썬 3(3.5 이상)을 사용한다
2. 라인 길이는 최대 100 문자
3. 이름 규칙(Naming)

-. Camelcased : 타입 , 클래스

-. snake_case : 파일, 패키지, 인터페이스, 모듈, 변수 , 함수 , 메소드

-. ALL_CAPITALS : 상수

4. 공백 문자 대 탭

-. 기본들여 쓰기(Indent)는 공백문자(Space) 4개를 사용한다(Tab 문자 사용 금지)

-. Hnaging indent(문장 중간에 들여쓰기를 사용하는 형식)의 사용법은

Class 의 접근 지정자 (public,protected,private)는 들여쓰기를 하지 않는다

5. 괄호(Brace)

-. if, else, do, while, for 구문에 괄호를 사용한다

6. 주석(Comments)

-. 문서 주석은”””” 을 사용한다

-. 구현 주석은 #을 사용한다

7. 린터(Linters)

-. 파이썬 코드 스타일의 자동 오류 검출을 위하여 amnet_flake8을 사용한다.

8. 기타

-. 모든 문자는 큰 따옴표(” , Double quotes)가 아닌 작은 따옴표(‘ , Single quote)를 사용하여 표현한다

ROS 프로그래밍 기초 (파이썬)

패키지 생성

사용자 작업 폴더

~/robot_ws/

shin@ubuntu:~/robot_ws/src$ ros2 pkg create **my_first_ros_rclpy_pkg(패키지 이름)** --build-type **ament_python(빌드 타입)** --dependencies **rclpy(의존 패키지)** **std_msgs(의존 패키지)**

Tree 구조 : 3 directories , 8 files

```
├── my_first_ros_rclpy_pkg
│   ├── my_first_ros_rclpy_pkg
│   │   └── __init__.py
│   ├── package.xml
│   ├── resource
│   │   └── my_first_ros_rclpy_pkg
│   ├── setup.cfg
│   ├── setup.py
│   └── test
│       ├── test_copyright.py
│       ├── test_flake8.py
│       └── test_pep257.py
```

패키지 설정

패키지 설정 파일 (Package.xml)

```xml
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematypens="http://-
www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_first_ros_rclpy_pkg</name>
  <version>0.0.0</version>
  <description>ROS2 rclpy basic package</description>
  <maintainer email="shinmk9476@gmail.com">shin</maintainer>
  <license>Apache License 2.0</license>

  <depend>rclpy</depend>
  <depend>std_msgs</depend>

  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>

  <export>
    <build_type>ament_python</build_type>
  </export>
</package>
```

파이썬 패키지 설정 파일 (setup.py)

-. entry_points 의 console_scripts를 사용하여 실행파일을 설정한다

helloworld_publisher / subcriber 은 my_first_ros_rclpy_pkg. helloworld_publisher / subscriber 모듈의 main 함수를 호출 하며 ros2 run / launch 명령어로 해당 스크립트를 실행.

```python
from setuptools import setup

package_name = 'my_first_ros_rclpy_pkg'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='shin',
    maintainer_email='shinmk9476@gmail.com',
    description='ROS2 rclpy basic package',
    license='Apache License, Version 2.0',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'helloworld_publisher = my_first_ros_rclpy_pkg.helloworld_publisher:main',
            'helloworld_subscriber = my_first_ros_rclpy_pkg.helloworld_subscriber:main',
        ],
    },
)
```

console script 에 Pub & Sub

파이썬 패키지 환경 설정 파일 (setup.cfg)

```
develop
script-dir=$base/lib/my_first_ros_rclpy_pkg
[install]
install-scripts=$base/lib/my_first_ros_rclpy_pkg
```

패키지 이름 기재 : my_first ~~~

colcon build 경우 /home/유저이름/robot_ws/
install/my_first_ros_rclpy_pkg/lib/my_first_ros_rclpy_pkg 폴더에 실행 파일 생성

퍼블리셔 노드 작성

import rclpy
from rclpy.node import Node
from rclpy.qos import QoSProfile
from std_msgs.msg import String

class HelloworldPublisher(Node):

```
def __init__(self):
    super().__init__('helloworld_publisher')
    qos_profile = QoSProfile(depth=10)
    self.helloworld_publisher = self.create_publisher(String, 'helloworld', qos_profil
e)

    self.timer = self.create_timer(1,self.publish_helloworld_msg)
    self.count = 0

def publish_helloworld_msg(self):
    msg = String()
    msg.data = 'Hello World: {0}' .format(self count)
    self.helloworld_publisher_.publish(msg)
    self.get_logger().info('Published message: {0}'.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()

서브스크라이브 노드 작성

import rclpy
from rclpy.node import Node
from rclpy.qos import QoSProfile
from std_msgs.msg import String

class HelloworldSubscriber(Node):

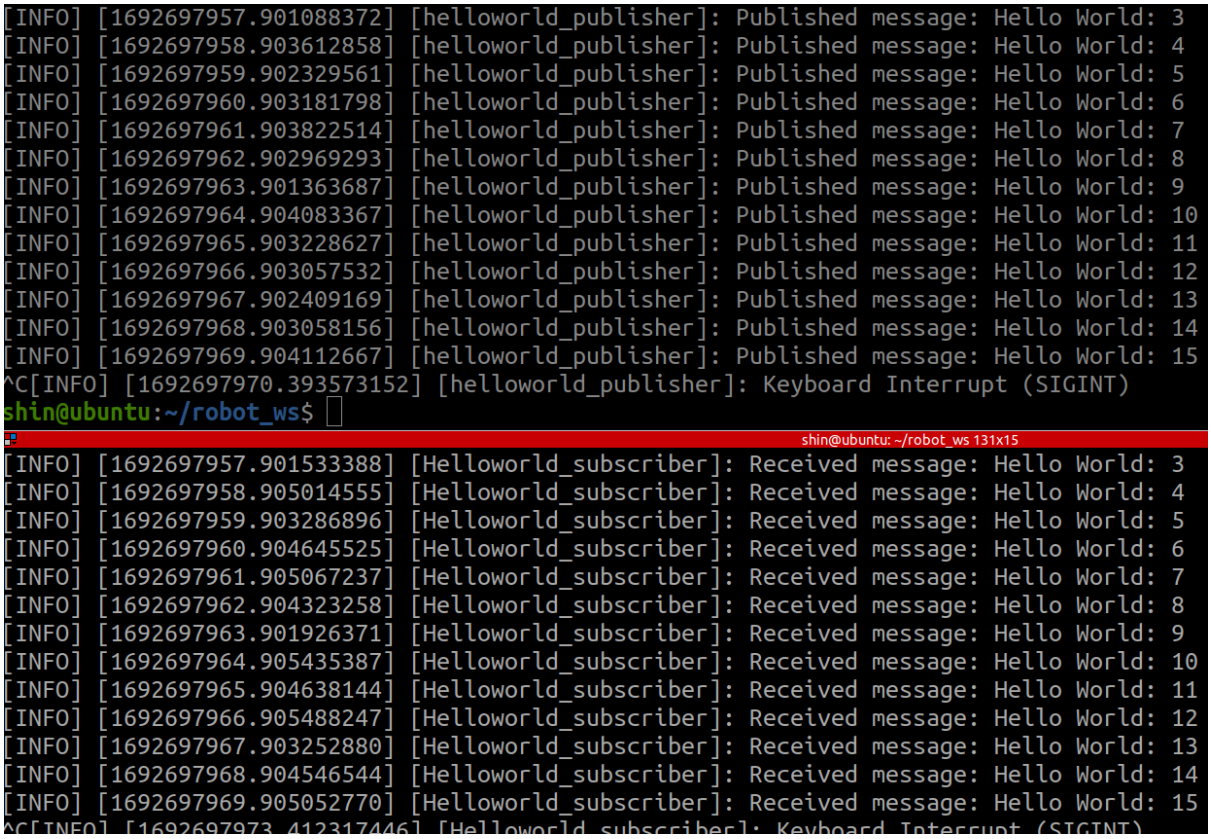
```
def __init__(self):
    super().__init__('Helloworld_subscriber')
    qos_profile = QoSProfile(depth=10)
    self.helloworld_subscriber = self.create_subscription(
        String,
        'helloworld',
        self.subscribe_topic_message,
        qos_profile)



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()

실행



ROS 프로그래밍 기초(C++)

패키지 생성

shin@ubuntu:~/robot_ws/src$ ros2 pkg create my_first_ros_rclcpp_pkg(패키지 이름) --build-type ament_cmake(빌드 타입) --dependencies rclcpp(의존 패키지) std_msgs(의존 패키지)

shin@ubuntu:~/robot_ws/src/my_first_ros_rclcpp_pkg$ tree

.**Tree 구조 : 3 directories , 2 files**

```
├── CMakeLists.txt
├── include
│   └── my_first_ros_rclcpp_pkg
├── package.xml
└── src
```

패키지 설정

패키지 설정 파일 (package . xml)

```xml
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematypens="http://-
  www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_first_ros_rclcpp_pkg</name>
  <version>0.0.0</version>
  <description>ROS2 rclcpp basic package</description>
  <maintainer email="shinmk9476@gmail.com">shin</maintainer>
  <license>Apache License 2.0</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <depend>rclcpp</depend>
  <depend>std_msgs</depend>

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

빌드 설정 파일 (CMakeLists.txt)

```
3
4 # Default to C99
5 if(NOT CMAKE_C_STANDARD)
6   set(CMAKE_C_STANDARD 99)
7 endif()
8
9 # Default to C++14
10 if(NOT CMAKE_CXX_STANDARD)
11   set(CMAKE_CXX_STANDARD 14)
12 endif()
13
14 if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
15   add_compile_options(-Wall -Wextra -Wpedantic)
16 endif()
17
18 # find dependencies
19 find_package(ament_cmake REQUIRED)
20 find_package(rclcpp REQUIRED)
21 find_package(std_msgs REQUIRED)
22
23 if(BUILD_TESTING)
24   find_package(ament_lint_auto REQUIRED)
25   # the following line skips the linter which checks for copyrights
26   # uncomment the line when a copyright and license is not present in all source files
27   #set(ament_cmake_copyright_FOUND TRUE)
28   # the following line skips cpplint (only works in a git repo)
29   # uncomment the line when this package is not in a git repo
30   #set(ament_cmake_cpplint_FOUND TRUE)
31   ament_lint_auto_find_test_dependencies()
32 endif()
33
34 # Build
35 add_executable(helloworld_publisher src/helloworld_publisher.cpp)
36 ament_target_dependencies(helloworld_publisher rclcpp std_msgs)
37
38 add_executable(helloworld_subscriber src/helloworld_subscriber.cpp)
39 ament_target_dependencies(helloworld_subscriber rclcpp std_msgs)
40
41 #Install
42 install(TARGETS
43   helloworld_publisher
44   helloworld_subscriber
45   DESTINATION lib/${PROJECT_NAME})
46
47 # Macro for ament package
48 ament_package()
```

퍼블리셔 노드 작성

```
#include <chrono>
#include <functional>
#include <memory>
#include <string>

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

using namespace std::chrono_literals;
```

```
class HelloworldPublisher : public rclcpp::Node
{
public:
HelloworldPublisher()
: Node("helloworld_publisher"), count_(0)
{
auto qos_profile = rclcpp::QoS(rclcpp::KeepLast(10));
helloworld_publisher_ = this->create_publisher<std_msgs::msg::String>(
"helloworld", qos_profile);
timer_ = this->create_wall_timer(
1s, std::bind(&HelloworldPublisher::publish_helloworld_msg, this));
}

private:
void publish_helloworld_msg()
{
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());
helloworld_publisher_->publish(msg);
}
rclcpp::TimerBase::SharedPtr timer_;
rclcpp::Publisher<std_msgs::msg::String>::SharedPtr helloworld_publisher_;
size_t count_;
};

int main(int argc, char * argv[])
{
rclcpp::init(argc, argv);
auto node = std::__make_shared<HelloworldPublisher>();
rclcpp::spin(node);
rclcpp::shutdown();
return 0;
}
```

서브스크라이버 노드

```
#include <functional>
#include <memory>
```

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

using std::placeholders::_1;

class HelloworldSubscriber : public rclcpp::Node
{
public:
HelloworldSubscriber()
: Node("minimal_subscriber")
{
auto qos_profile = rclcpp::QoS(rclcpp::KeepLast(10));
helloworld_subscriber_ = this->create_subscription<std_msgs::msg::String>(
"helloworld", qos_profile,
std::bind(&HelloworldSubscriber::subscribe_topic_message, this, _1));
}

private:
void subscribe_topic_message(const std_msgs::msg::String::SharedPtr msg) const
{
RCLCPP_INFO(this->get_logger(), "Received message: '%s'", msg->data.c_str());
}
rclcpp::Subscription<std_msgs::msg::String>::SharedPtr helloworld_subscriber_;
};

int main(int argc, char * argv[])
{
rclcpp::init(argc, argv);
auto node = std::make_shared<HelloworldSubscriber>();
rclcpp::spin(node);
rclcpp::shutdown();
return 0;
}
```

ROS 2 Tips

설정 스크립트 (setup script)

$ source ~/robot_ws/install/local_setup.bash

$ source ~/opt/ros/foxy/setup.bash

ROS Package / node 구동 시키기 위한 설정 스크립트

setup.bash vs local_setup.bash

Underlay	Ros 설치시 자동 다운 되는 위치 (opt/ros/foxy) & 소스 코드 다운/빌드 되는 위치 ros2-foxy
Overlay	사용자가 설정한 작업 공간 (robot_ws & test_ws) 으로 Underlay의 소스 사용으로 의존성
setup.bash	작업 공간 이외에 환경 설정에 연관된 다른 공간까지 빌드 및 설정
local_setup.bash	해당 작업 공간에 한해서 진행

ROS_DOMAIN_ID vs Namespace

ROS_DOMAIN_ID

ROS2 에서는 DDS Global space를 쉽게 변경 하는 방법으로 ROS_DOMAIN_ID 환경 변수를 두고 있고 각 RMW 에서는 이 환경변수를 참조하여 도메인을 바꾼다

ROS_DOMAIN_ID=30 만 통신, ID=20 listener는 통신 불가 확인 및 ID는 RMW 에서 정수 0 ~ 101 까지 사용.

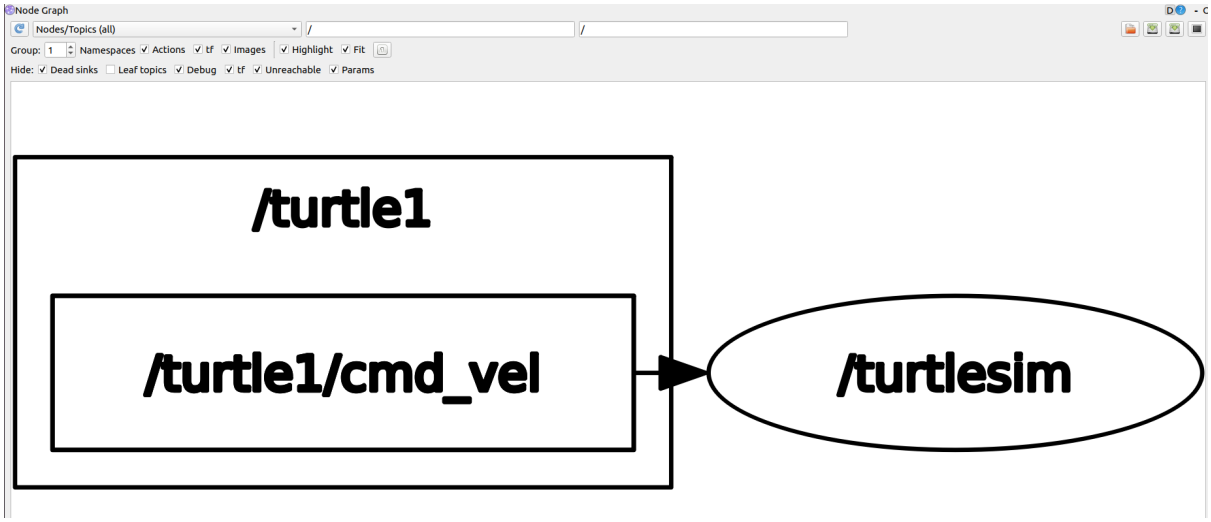
ROS2 Namespace

ROS Node 생성 방법

1. Ros 명령어에 ns(namespace)를 입력하는 방법
2. lunch 파일로 실행시 node_namespacce를 사용 하는 방법

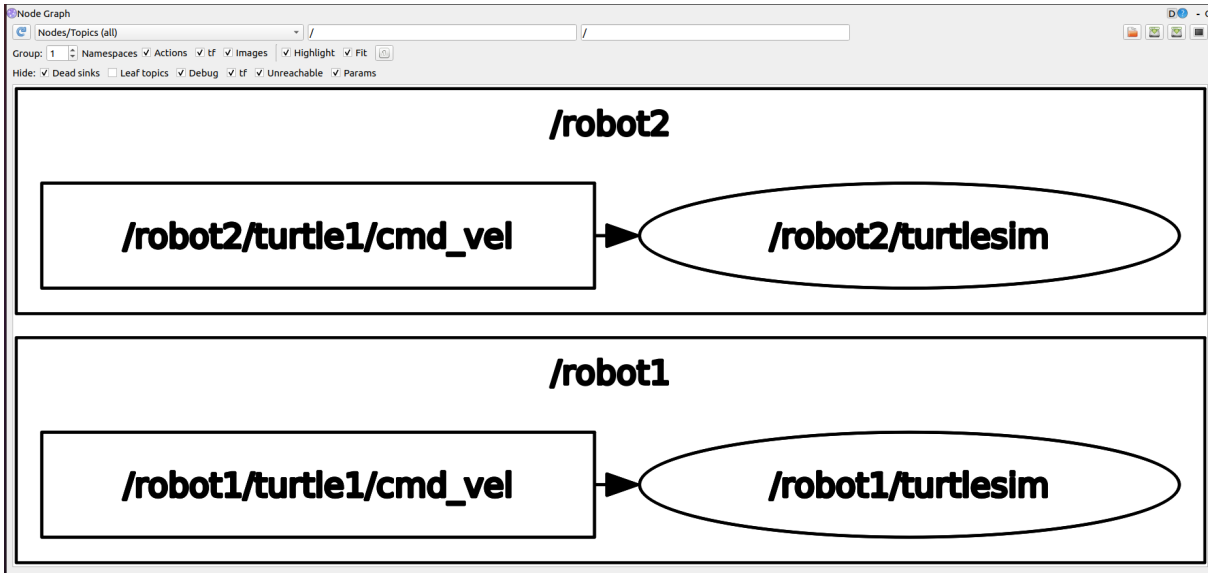
Ros 명령어에 ns(namespace)를 입력하는 방법

shin@ubuntu:~$ ros2 run turtlesim turtlesim_node



ros2 run turtlesim turtlesim_node --ros-args -r __ns:=/robot1

ros2 run turtlesim turtlesim_node --ros-args -r __ns:=/robot2



토픽, 서비스, 액션 인터페이스

인터페이스 패키지 생성 및 msg, srv, action 폴더 생성 및 인터페이스 파일 작성

$ cd ~/robot_ws/src

$ ros2 pkg create - -build-type ament_cmake msg_srv_action_interface_example

$ cd msg_srv_action_interface_example : 패키지 이동

$ mkdir msg srv action : 3 종류 폴더 생성

shin@ubuntu:~/robot_ws/src/msg_srv_action_interface_example$ tree : tree 구조

```
.
├── action
│   └── ArithmeticChecker.action
├── CMakeLists.txt
├── include
│   └── msg_srv_action_interface_example
├── msg
│   └── ArithmeticArgument.msg
├── package.xml
├── src
└── srv
    └── ArithmeticOperator.srv
```

ArithmeticArgument.msg (msg file)

Messages

builtin_interfaces/Time stamp
float32 argument_a
float32 argument_b

ArithmeticOperator.srv

Constants

int8 PLUS = 1
int8 MINUS = 2
int8 MULTIPLY = 3
int8 DIVISION = 4

Request

int8 arithmetic_operator

— - - -

Response

float32 arithmetic_result

ArithmeticChecker.action

Goal

float32 goal_sum

— - - -

Result

string[] all_formula
float32 total_sum

— - - -

Feedback

string[] formula

package.xml

```xml
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>msg_srv_action_interface_example</name>
  <version>0.0.0</version>
  <description>ROS2 example for message</description>
  <maintainer email="shinmk9476@gmail.com">shin</maintainer>
  <license>Apache License 2.0</license>

  <buildtool_depend>ament_cmake</buildtool_depend>
  <buildtool_depend>rosidl_default_generators</buildtool_depend>
  <exec_depend>builtin_interfaces</exec_depend>
  <exec_depend>rosidl_default_runtime</exec_depend>
  <member_of_group>rosidl_interface_packages</member_of_group>

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

```
18 # find dependencies
19 find_package(ament_cmake REQUIRED)
20 find_package(builtin_interfaces REQUIRED)
21 find_package(rosidl_default_generators REQUIRED)
22 # uncomment the following section in order to fill in
23 # further dependencies manually.
24 # find_package(<dependency> REQUIRED)
25
26 if(BUILD_TESTING)
27   find_package(ament_lint_auto REQUIRED)
28   # the following line skips the linter which checks for copyrights
29   # uncomment the line when a copyright and license is not present in all source files
30   #set(ament_cmake_copyright_FOUND TRUE)
31   # the following line skips cpplint (only works in a git repo)
32   # uncomment the line when this package is not in a git repo
33   #set(ament_cmake_cpplint_FOUND TRUE)
34   ament_lint_auto_find_test_dependencies()
35 endif()
36
37 # Declare ROS messages, services and actions
38
39 set(msg_files
40   "msg/ArithmeticArgument.msg")
41 set(srv_files
42   "srv/ArithmeticOperator.srv")
43 set(action_files
44   "action/ArithmeticChecker.action")
45
46 rosidl_generate_interfaces(${PROJECT_NAME}
47   ${msg_files}
48   ${srv_files}
49   ${action_files}
50   DEPENDENCIES builtin_interfaces)
51
52 # Macro for ament package
53 ament_export_dependencies(rosidl_default_runtime)
54 ament_package()
```

shin@ubuntu:~/robot_ws/src/msg_srv_action_interface_example$ tree

```
.
├── action
│   └── ArithmeticChecker.action
├── CMakeLists.txt
├── include
│   └── msg_srv_action_interface_example
├── msg
│   └── ArithmeticArgument.msg
├── package.xml
├── src
└── srv
    └── ArithmeticOperator.srv
```