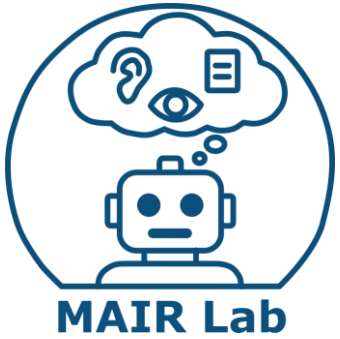
ROS2: Creating custom msg and srv files

운영체제의 실제

안인규 (Inkyu An)



Create custom message type

- In the previous lecture, we practiced how to use Python for Topic communication.
- At that time, we used the predefined *std_msgs/msg/String* type.
- But what if we need a new topic type?

```python
import rclpy
from rclpy.node import Node

from std_msgs.msg import String
```

ros2 pkg prefix std_msgs

/opt/ros/humble
(ROS2 설치 시, 함께 설치되는 standard message type)

```python
class MinimalPublisher(Node):

    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 0.5  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0

    def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1
```

Create custom message type

- To create a custom message type, you must use *ament_cmake*.
- You can generate a new message type by following the steps below.

1. Create a package (ament_cmake)
2. Make a "*.msg" or "*.srv" file
3. Modify "package.xml", "CMakeLists.txt"
4. Build

- Create a new package for creating custom message types:

```
$ ros2 pkg create --build-type ament_cmake --license Apache-2.0 custom_msg_pkg
```

Create custom message type

1. **Create a package (ament_cmake)**
2. **Make a "*.msg" or "*.srv" file**
3. Modify "package.xml", "CMakeLists.txt"
4. Build

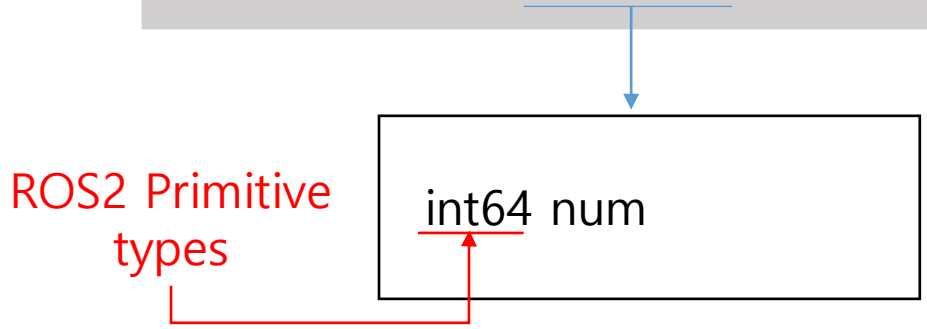
- Then, we have to create the directories in *ros2_ws/src/custom_msg_pkg* :

```
$ mkdir msg srv
```

  - **msg**: topic message types
  - **srv**: service message types

- Go to the *msg* directory, and create a *Num.msg* file

```
$ cd msg; vim Num.msg
```



Create custom message type

1. Create a package (ament_cmake)
2. **Make a "*.msg" or "*.srv" file**
3. Modify "package.xml", "CMakeLists.txt"
4. Build

- ROS2 Primitive data types

Type name	C++	Python	DDS type
bool	bool	builtins.bool	boolean
byte	uint8_t	builtins.bytes*	octet
char	char	builtins.str*	char
float32	float	builtins.float*	float
float64	double	builtins.float*	double
int8	int8_t	builtins.int*	octet
uint8	uint8_t	builtins.int*	octet
int16	int16_t	builtins.int*	short
uint16	uint16_t	builtins.int*	unsigned short
int32	int32_t	builtins.int*	long
uint32	uint32_t	builtins.int*	unsigned long
int64	int64_t	builtins.int*	long long
uint64	uint64_t	builtins.int*	unsigned long long
string	std::string	builtins.str	string
wstring	std::u16string	builtins.str	wstring

Primitive data types을 자유롭게 사용 가능
그렇다면, array type도 가능한가?

Create custom message type

1. Create a package (ament_cmake)
2. **Make a "*.msg" or "*.srv" file**
3. Modify "package.xml", "CMakeLists.txt"
4. Build

- ROS2 Primitive data types

Type name	C++	Python	DDS type
bool	bool	builtins.bool	boolean
byte	uint8_t	builtins.bytes*	octet
char	char	builtins.str*	char
float32	float	builtins.float*	float
float64	double	builtins.float*	double
int8	int8_t	builtins.int*	octet
uint8	uint8_t	builtins.int*	octet
int16	int16_t	builtins.int*	short
uint16	uint16_t	builtins.int*	unsigned short
int32	int32_t	builtins.int*	long
uint32	uint32_t	builtins.int*	unsigned long
int64	int64_t	builtins.int*	long long
uint64	uint64_t	builtins.int*	unsigned long long
string	std::string	builtins.str	string
wstring	std::u16string	builtins.str	wstring

Every Built-in-type can be used to define arrays:

Type name	C++	Python	DDS type
static array	std::array<T, N>	builtins.list*	T[N]
unbounded dynamic array	std::vector	builtins.list	sequence
bounded dynamic array	custom_class<T, N>	builtins.list*	sequence<T, N>
bounded string	std::string	builtins.str*	string

- **static array**:
  int8[10] var (길이가 10인 배열)
- **unbounded dynamic array**:
  int8[] var (길이가 가변인 배열)
- **bounded dynamic array**:
  int8[<=5] var (길이가 5개 이하인 배열)
- **bounded string**:
  string<=10 name (문자가 10개 이하인 문자열)

Create custom message type

1. Create a package (ament_cmake)
2. **Make a "*.msg" or "*.srv" file**
3. Modify "package.xml", "CMakeLists.txt"
4. Build

- What if you want to use a type that is not included in the ROS 2 primitive data types?

```
$ cd msg; vim RobotPos.msg
```

geometry_msgs: packag를 build할 때, 참고해야 하는 dependenc가 되며, package.xml과 CMakeLists.txt 파일에서 정의가 되어야함

Create custom message type

1. Create a package (ament_cmake)
2. Make a "*.msg" or "*.srv" file
3. **Modify "package.xml", "CMakeLists.txt"**
4. Build

- How to modify '**package.xml**' and '**CMakeLists.txt**' files when using C++ (the talker and listener example)
- Package.xml:
  1. Meta information:

```
<description>Examples of minimal publisher/subscriber using rclcpp</description>
<maintainer email="you@email.com">Your Name</maintainer>
<license>Apache License 2.0</license>
```

  2. Add dependencies:

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

Create custom message type

1. Create a package (ament_cmake)
2. Make a "*.msg" or "*.srv" file
3. **Modify "package.xml", "CMakeLists.txt"**
4. Build

- How to modify '**package.xml**' and '**CMakeLists.txt**' files when using C++ (the talker and listener example)
- CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.5)
project(cpp_pubsub)

# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)
```

```
add_executable(talker src/publisher_member_function.cpp)
ament_target_dependencies(talker rclcpp std_msgs)

add_executable(listener src/subscriber_member_function.cpp)
ament_target_dependencies(listener rclcpp std_msgs)

install(TARGETS
  talker
  listener
  DESTINATION lib/${PROJECT_NAME})

ament_package()
```

Create custom message type

1. Create a package (ament_cmake)
2. Make a "*.msg" or "*.srv" file
3. **Modify "package.xml", "CMakeLists.txt"**
4. Build

```
...
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)

add_executable(talker src/publisher_member_function.cpp)
ament_target_dependencies(talker rclcpp std_msgs)



add_executable(listener src/subscriber_member_function.cpp)
ament_target_dependencies(listener rclcpp std_msgs)

install(TARGETS
  talker
  listener
  DESTINATION lib/${PROJECT_NAME})
...
```

Create custom message type

1. Create a package (ament_cmake)
2. Make a "*.msg" or "*.srv" file
3. **Modify "package.xml", "CMakeLists.txt"**
4. Build

```
...
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)

add_executable(talker src/publisher_member_function.cpp)
ament_target_dependencies(talker rclcpp std_msgs)

add_executable(listener src/subscriber_member_function.cpp)
ament_target_dependencies(listener rclcpp std_msgs)

install(TARGETS
  talker
  listener
  DESTINATION lib/${PROJECT_NAME})
...
```

Add dependencies:

- ament_cmake: ROS2 build system package
- rclcpp: ROS2 C++ client 라이브러리
- std_msgs: Standard message type

talker node:

- publisher_memer_function.cpp를 컴파일해 talker 실행 파일 생성 (/build/..)
- rclcpp, std_msgs 라이브러리와 링크

listener node:

- subscriber_memer_function.cpp를 컴파일해 listener 실행 파일 생성
- rclcpp, std_msgs 라이브러리와 링크

설치 경로 지정:

- 빌드된 실행파일 talker와 listener를 설치
- install/... 에 설치

Create custom message type

1. Create a package (ament_cmake)
2. Make a "*.msg" or "*.srv" file
3. **Modify "package.xml", "CMakeLists.txt"**
4. Build

- How to modify '**package.xml**' and '**CMakeLists.txt**' files to create custom message types
- package.xml:
  1. Meta information:

```
<description>Examples of minimal publisher/subscriber using rclcpp</description>
<maintainer email="you@email.com">Your Name</maintainer>
<license>Apache License 2.0</license>
```

  2. Add dependencies:

```
<depend>geometry_msgs</depend>
<buildtool_depend>rosidl_default_generators</buildtool_depend>
<exec_depend>rosidl_default_runtime</exec_depend>
<member_of_group>rosidl_interface_packages</member_of_group>
```

Create custom message type

1. Create a package (ament_cmake)
2. Make a "*.msg" or "*.srv" file
3. **Modify "package.xml", "CMakeLists.txt"**
4. Build

- How to modify '**package.xml**' and '**CMakeLists.txt**' files to create custom message types
- package.xml:
  1. Meta information:

```
<description>Examples of minimal publisher/subscriber using rclcpp</description>
<maintainer email="you@email.com">Your Name</maintainer>
<license>Apache License 2.0</license>
```

  2. Add dependencies:

```
<depend>geometry_msgs</depend>
<buildtool_depend>rosidl_default_generators</buildtool_depend>
<exec_depend>rosidl_default_runtime</exec_depend>
<member_of_group>rosidl_interface_packages</member_of_group>
```

빌드와 실행할 때 모두 필요

해당 package를 빌드할 때, 필요한 tool 명시

실행에서만 필요

해당 package를 특정 dependency group에 속하게 함; 즉, ROS2 interface (topic, service)를 제공하는 package라고 명시

Create custom message type

1. Create a package (ament_cmake)
2. Make a "*.msg" or "*.srv" file
3. **Modify "package.xml", "CMakeLists.txt"**
4. Build

- How to modify '**package.xml**' and '**CMakeLists.txt**' files to create custom message types
- CMakeLists.txt:

```
find_package(geometry_msgs REQUIRED)
find_package(rosidl_default_generators REQUIRED)

rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/Num.msg"
  "msg/RobotPos.msg"
  DEPENDENCIES geometry_msgs # Add packages that above messages depend on,
in this case geometry_msgs for RobotPos.msg
)
```

topic과 service에 대해서 각 언어별 코드 (C++, Python 등)을 자동으로 생성

Build

1. Create a package (ament_cmake)
2. Make a “*.msg” or “*.srv” file
3. Modify “package.xml”, “CMakeLists.txt”
4. **Build**

- Build the package:

```
$ colcon build --packages-select custom_msg_pkg
```

- Overlay install/setup.bash:

```
$ source install/setup.bash
```

- Check the custom message type:

```
$ ros2 interface show custom_msg_pkg/msg/Num
```

```
$ ros2 interface show custom_msg_pkg/msg/RobotPos
```

Utilize custom message type

- Modify your “py_pubsub” package to utilize custom message type “custom_msg_pkg/msg/Num”
  - Hint!

package.xml:

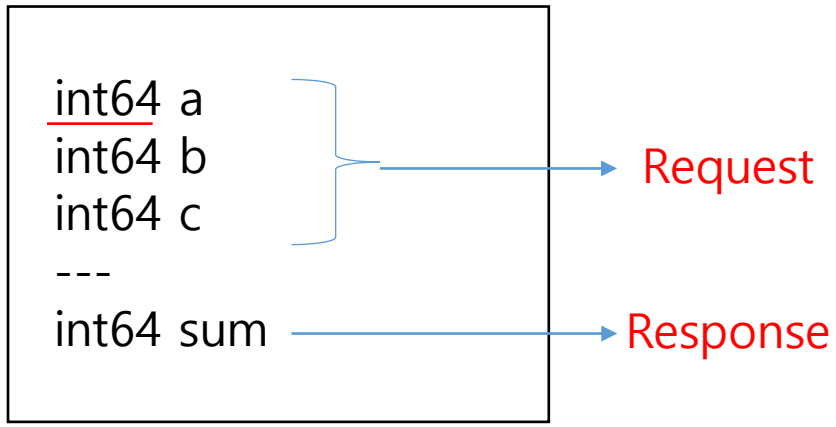
```
<exec_depend> custom_msg_pkg </exec_depend>
```

Create custom message type

1. **Create a package (ament_cmake)**
2. **Make a "*.msg" or "*.srv" file**
3. Modify "package.xml", "CMakeLists.txt"
4. Build

- Go to the *srv* directory, and create a *Num.msg* file

```
$ cd srv; vim AddThreeInts.srv
```



Create custom message type

1. Create a package (ament_cmake)
2. Make a "*.msg" or "*.srv" file
3. **Modify "package.xml", "CMakeLists.txt"**
4. Build

- How to modify '**package.xml**' and '**CMakeLists.txt**' files to create custom message types
- CMakeLists.txt:

```
find_package(geometry_msgs REQUIRED)
find_package(rosidl_default_generators REQUIRED)

rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/Num.msg"
  "msg/robot_pos.msg"
  "srv/AddThreeInts.srv"
  DEPENDENCIES geometry_msgs # Add packages that above messages depend on,
in this case geometry_msgs for robot_pos.msg
)
```

Utilize custom message type

- Create the nodes that utilize the custom service type (custom_msg_pkg/srv/AddThreeInts)
  - Service node: service_member_function.py
  - Client node: client_member_function.py

- Modify “package.xml” and “setup.py”

package.xml:

```
<exec_depend>custom_msg_pkg</exec_depend>
```

setup.py:

```
entry_points={
    'console_scripts': [
        ...
        'service = py_srvcli.service_member_function:main',
        'client = py_srvcli.client_member_function:main',
    ],
},
```