2부 ROS 2 기본 프로그래밍
6장 ~ 11장 (파이썬)

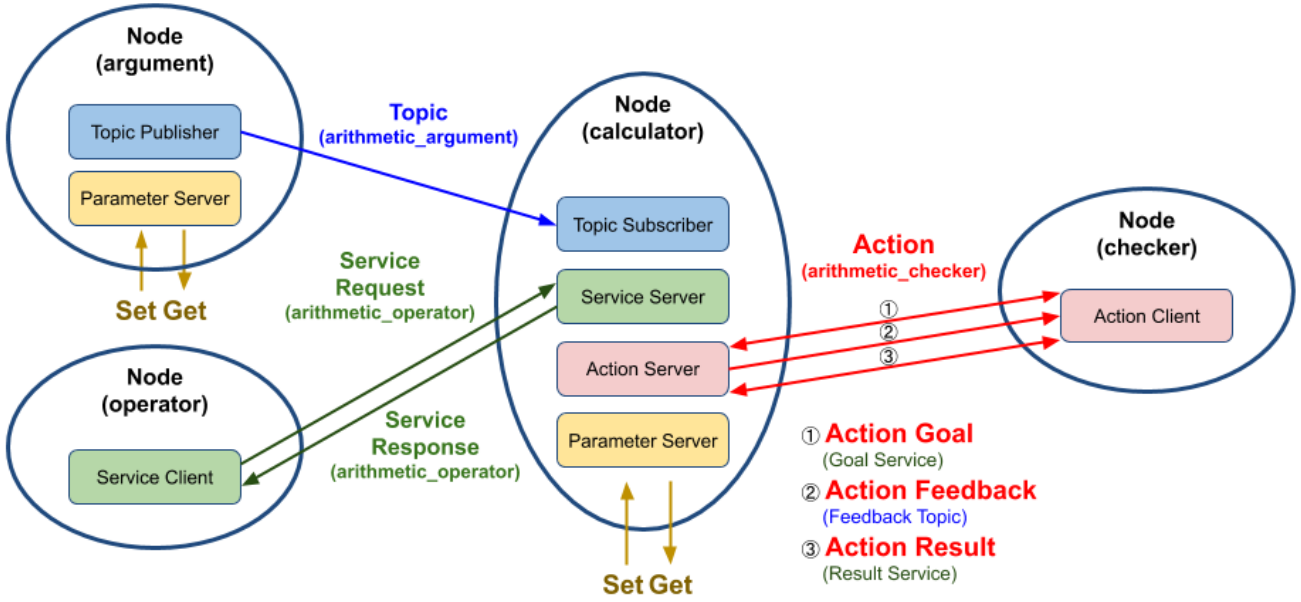
최현진

목차

6장 ROS2 패키지 설계 (파이썬)

6.1. 패키지 설계

- topic_service_action_rclpy_example 패키지
  - 4개의 Node 로 구성



(6.5. 소스코드 내려받기 및 빌드)

```
jasmine@jasmine-VirtualBox:~/robot_ws/src$ git clone https://github.com/robotpilot/ros
2-seminar-examples.git
Cloning into 'ros2-seminar-examples'...
remote: Enumerating objects: 1284, done.
remote: Counting objects: 100% (733/733), done.
remote: Compressing objects: 100% (386/386), done.
remote: Total 1284 (delta 411), reused 552 (delta 278), pack-reused 551
Receiving objects: 100% (1284/1284), 256.84 KiB | 519.00 KiB/s, done.
Resolving deltas: 100% (689/689), done.
jasmine@jasmine-VirtualBox:~/robot_ws/src$
```

```
jasmine@jasmine-VirtualBox:~/robot_ws$ colcon build --symlink-install
Starting >>> msg_srv_action_interface_example
Finished <<< msg_srv_action_interface_example [29.4s]
Starting >>> logging_rclpy_example
Finished <<< logging_rclpy_example [1.54s]
Starting >>> my_first_ros_rclcpp_pkg
Finished <<< my_first_ros_rclcpp_pkg [17.7s]
Starting >>> my_first_ros_rclpy_pkg
Finished <<< my_first_ros_rclpy_pkg [1.43s]
Starting >>> ros2env
Finished <<< ros2env [1.44s]
Starting >>> rqt_example
Finished <<< rqt_example [2.06s]
```

6.3. 패키지 설정 파일 (package.xml)

- topic_service_action_rclpy_example 패키지의 설정 파일

package.xml

src > ros2-seminar-examples > topic_service_action_rclpy_example > package.xml

```
1  <?xml version="1.0"?>
2  <?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematypens="http://www.w3.org/2001/XMLSchema"?>
3  <package format="3">
4    <name>topic_service_action_rclpy_example</name>
5    <version>0.6.0</version>
6    <description>ROS 2 rclpy example package for the topic, service, action</description>
7    <maintainer email="passionvirus@gmail.com">Pyo</maintainer>
8    <license>Apache License 2.0</license>
9    <author email="passionvirus@gmail.com">Pyo</author>
10   <author email="routiful@gmail.com">Darby Lim</author>
11   <depend>rclpy</depend>
12   <depend>std_msgs</depend>
13   <depend>msg_srv_action_interface_example</depend>
14   <test_depend>ament_copyright</test_depend>
15   <test_depend>ament_flake8</test_depend>
16   <test_depend>ament_pep257</test_depend>
17   <test_depend>python3-pytest</test_depend>
18   <export>
19     <build_type>ament_python</build_type>
20   </export>
21 </package>
```

← 2부 5장에서 작성한 인터페이스 패키지

6.4. 파이썬 패키지 설정 파일 (setup.py)

src > ros2-seminar-examples > topic_service_action_rclpy_example > setup.py > ...

```python
#!/usr/bin/env python3

import glob
import os

from setuptools import find_packages
from setuptools import setup

package_name = 'topic_service_action_rclpy_example'
share_dir = 'share/' + package_name

setup(
    name=package_name,
    version='0.6.0',
    packages=find_packages(exclude=['test']),
    data_files=[
        ('share/ament_index/resource_index/packages', ['resource/' + package_name]),
        (share_dir, ['package.xml']),
        (share_dir + '/launch', glob.glob(os.path.join('launch', '*.launch.py'))),
        (share_dir + '/param', glob.glob(os.path.join('param', '*.yaml'))),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    author='Pyo, Darby Lim',
    author_email='passionvirus@gmail.com, routiful@gmail.com',
    maintainer='Pyo',
    maintainer_email='passionvirus@gmail.com',
    keywords=['ROS'],
    classifiers=[
        'Intended Audience :: Developers',
        'License :: OSI Approved :: Apache Software License',
        'Programming Language :: Python',
        'Topic :: Software Development',
    ],
    description='ROS 2 rclpy example package for the topic, service, action',
    license='Apache License, Version 2.0',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'argument = topic_service_action_rclpy_example.arithmetic.argument:main',
            'operator = topic_service_action_rclpy_example.arithmetic.operator:main',
            'calculator = topic_service_action_rclpy_example.calculator.main:main',
            'checker = topic_service_action_rclpy_example.checker.main:main',
        ],
    },
)
```

빌드 후에 설치 폴더에 생성되는 파일들

```python
data_files=[
    ('share/ament_index/resource_index/packages', ['resource/' + package_name]),
    (share_dir, ['package.xml']),
    (share_dir + '/launch', glob.glob(os.path.join('launch', '*.launch.py'))),
    (share_dir + '/param', glob.glob(os.path.join('param', '*.yaml'))),
],
```

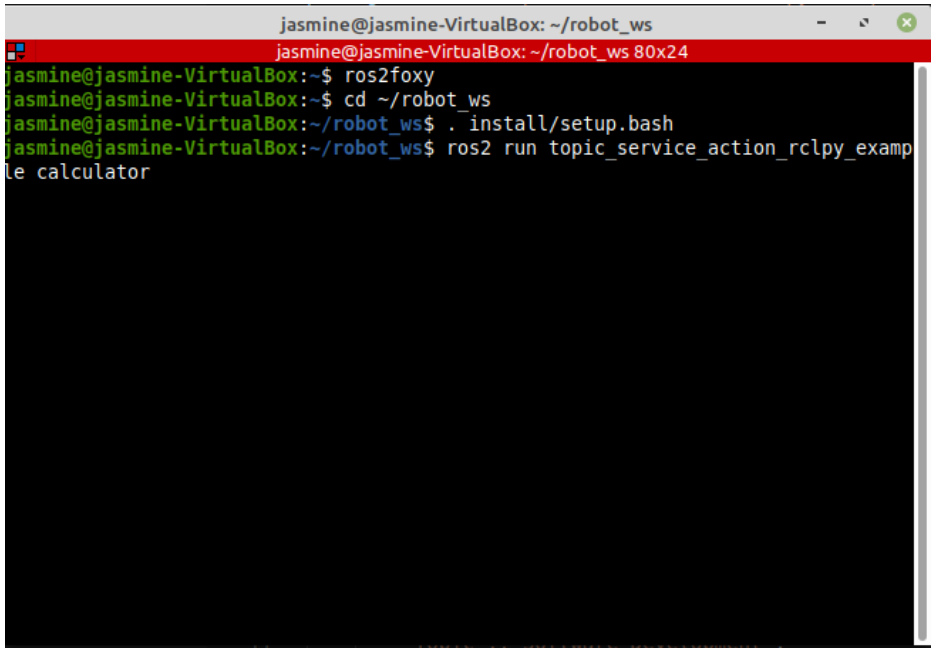
각 노드의 실행 명령어로 실행시키는 스크립트

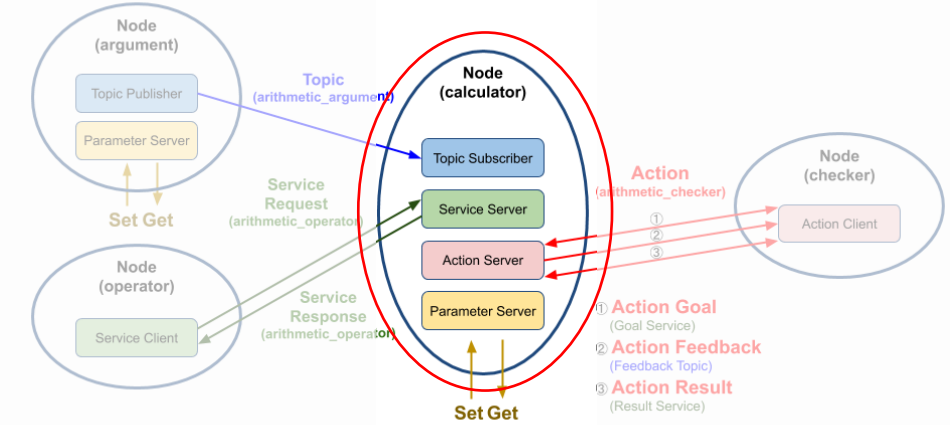
```python
entry_points={
    'console_scripts': [
        'argument = topic_service_action_rclpy_example.arithmetic.argument:main',
        'operator = topic_service_action_rclpy_example.arithmetic.operator:main',
        'calculator = topic_service_action_rclpy_example.calculator.main:main',
        'checker = topic_service_action_rclpy_example.checker.main:main',
    ],
},
```

excutables name

6.6. 실행

- 6.6.1 calculator node
  - Topic subscriber
  - Service server
  - Action server



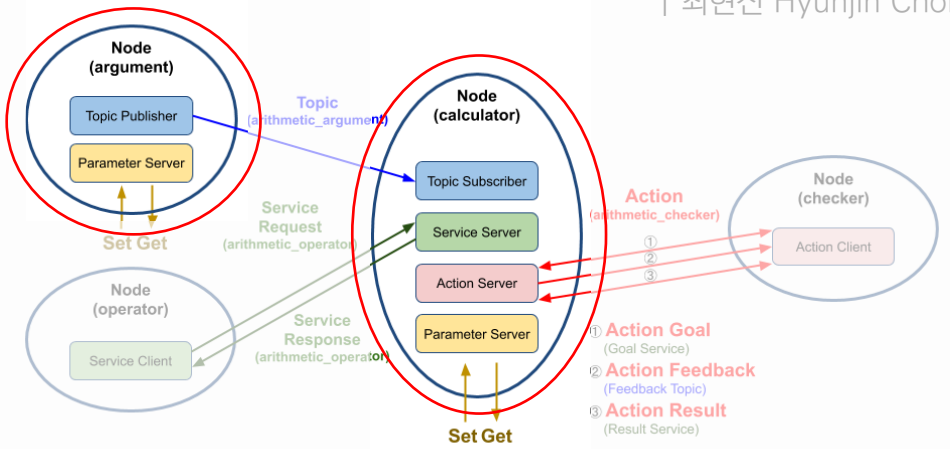


6.6. 실행

- 6.6.2 argument node
  - Topic publisher

Topic name: arithmatic_argument
Interface: ArithmeticArgument.msg

```
# Messages
builtin_interfaces/Time stamp
float32 argument_a
float32 argument_b
```



calculator

argument

jasmine@jasmine-VirtualBox: ~/robot_ws

```
jasmine@jasmine-VirtualBox:~/robot_ws$ . install/setup.bash
jasmine@jasmine-VirtualBox:~/robot_ws$ ros2 run topic_service_action_rclpy
_example calculator
[INFO] [1634839777.655924081] [calculator]: Timestamp of the message: buil
tin_interfaces.msg.Time(sec=1634839777, nanosec=506642329)
[INFO] [1634839777.656660102] [calculator]: Subscribed argument a: 4.0
[INFO] [1634839777.657351893] [calculator]: Subscribed argument b: 8.0
[INFO] [1634839778.512410436] [calculator]: Timestamp of the message: buil
tin_interfaces.msg.Time(sec=1634839778, nanosec=506480768)
[INFO] [1634839778.534703738] [calculator]: Subscribed argument a: 0.0
[INFO] [1634839778.538375015] [calculator]: Subscribed argument b: 2.0
[INFO] [1634839779.511531910] [calculator]: Timestamp of the message: buil
tin_interfaces.msg.Time(sec=1634839779, nanosec=506283946)
[INFO] [1634839779.518610526] [calculator]: Subscribed argument a: 7.0
[INFO] [1634839779.528158579] [calculator]: Subscribed argument b: 0.0
[INFO] [1634839780.512190301] [calculator]: Timestamp of the message: buil
tin_interfaces.msg.Time(sec=1634839780, nanosec=506516778)
[INFO] [1634839780.536408291] [calculator]: Subscribed argument a: 4.0
[INFO] [1634839780.557886747] [calculator]: Subscribed argument b: 0.0
[INFO] [1634839781.512802573] [calculator]: Timestamp of the message: buil
tin_interfaces.msg.Time(sec=1634839781, nanosec=506561788)
[INFO] [1634839781.534210080] [calculator]: Subscribed argument a: 4.0
[INFO] [1634839781.537565779] [calculator]: Subscribed argument b: 6.0
```

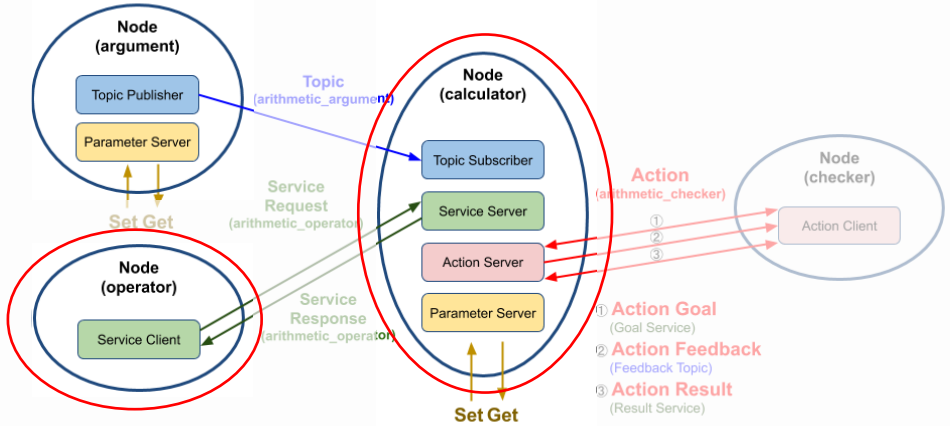
```
jasmine@jasmine-VirtualBox:~/robot_ws$ ros2foxy
jasmine@jasmine-VirtualBox:~/robot_ws$ . install/setup.bash
jasmine@jasmine-VirtualBox:~/robot_ws$ ros2 run topic_service_action_rclpy_
example argument
[INFO] [1634839777.622910398] [argument]: Published argument a: 4.0
[INFO] [1634839777.626324392] [argument]: Published argument b: 8.0
[INFO] [1634839778.518981524] [argument]: Published argument a: 0.0
[INFO] [1634839778.535519620] [argument]: Published argument b: 2.0
[INFO] [1634839779.512602628] [argument]: Published argument a: 7.0
[INFO] [1634839779.520313060] [argument]: Published argument b: 0.0
[INFO] [1634839780.516883173] [argument]: Published argument a: 4.0
[INFO] [1634839780.536143212] [argument]: Published argument b: 0.0
[INFO] [1634839781.516898822] [argument]: Published argument a: 4.0
[INFO] [1634839781.538950005] [argument]: Published argument b: 6.0
```

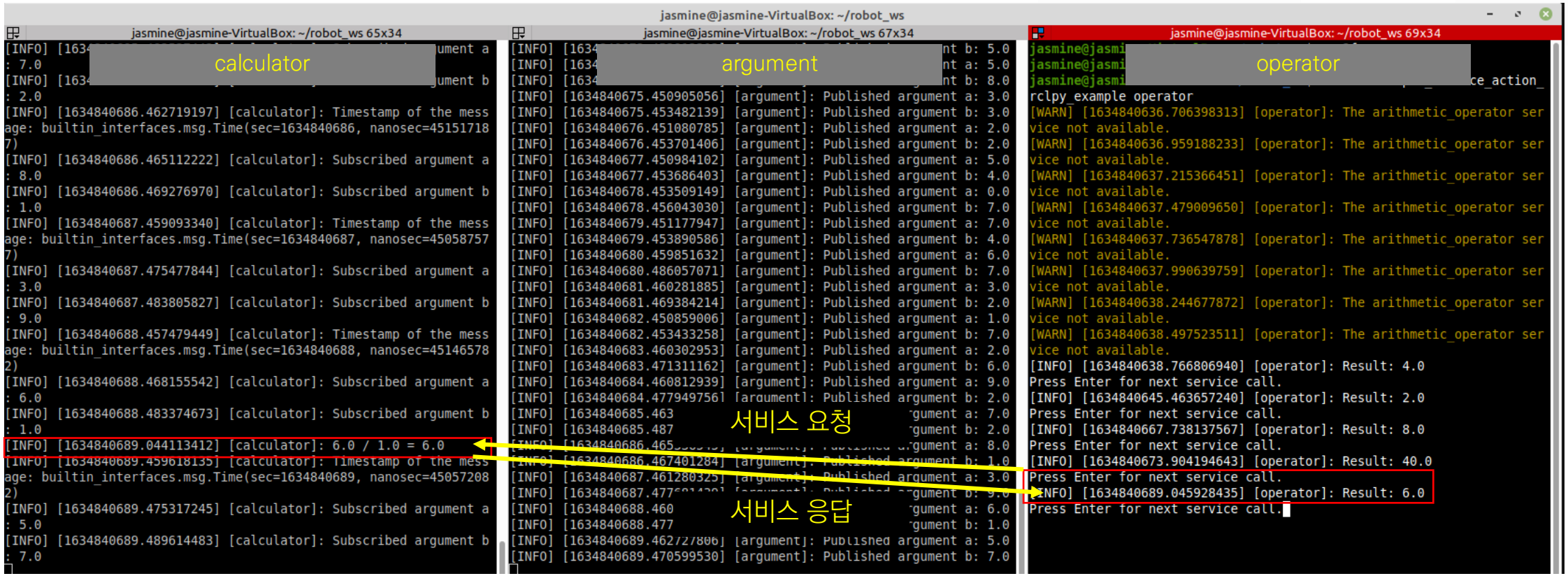
6.6. 실행

- 6.6.3 operator node
  - Service client

Interface:
ArithmeticOperator.srv

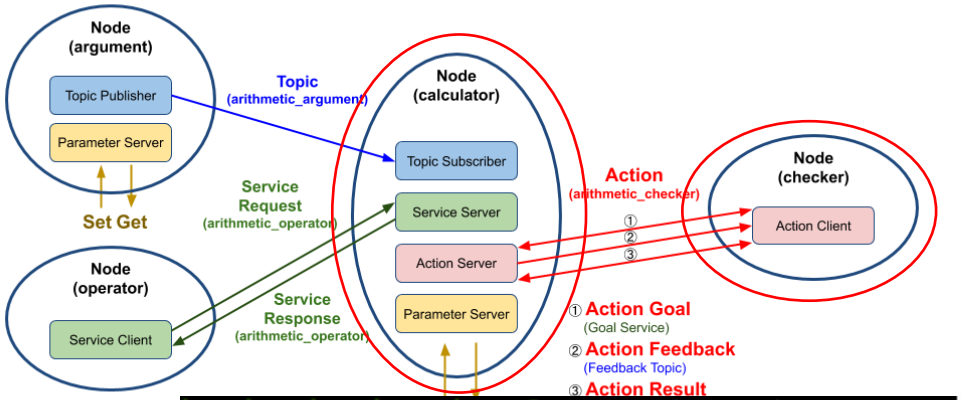
```
# Constants
int8 PLUS = 1
int8 MINUS = 2
int8 MULTIPLY = 3
int8 DIVISION = 4

# Request
int8 arithmetic_operator
---
# Response
float32 arithmetic_result
```





6.6. 실행

- 6.6.4 checker node
  - Action client



jasmine@jasmine-VirtualBox: ~/robot_ws

jasmine@jasmine-VirtualBox: ~/robot_ws 70x34

calculator

```
[INFO] [1634841565.455219286] [calculator]: Subscribed argument b: 3.0
[INFO] [163                                                      .0
[INFO] [163                                                      0 + 2.0 = 5.
0', '6.0 * 2.0 = 12.0', '6.0 * 2.0 = 12.0', '9.0 - 9.0 = 0.0', '9.0 -
9.0 = 0.0', '9.0 + 3.0 = 12.0']
[INFO] [1634841566.462016404] [calculator]: Timestamp of the message:
builtin_interfaces.msg.Time(sec=1634841566, nanosec=450621626)
[INFO] [1634841566.464788060] [calculator]: Subscribed argument a: 3.0
[INFO] [1634841566.489179839] [calculator]: Subscribed argument b: 6.0
[INFO] [1634841567.209294715] [calculator]: Feedback: ['3.0 + 2.0 = 5.
0', '6.0 * 2.0 = 12.0', '6.0 * 2.0 = 12.0', '9.0 - 9.0 = 0.0', '9.0 -
9.0 = 0.0', '9.0 + 3.0 = 12.0', '9.0 + 3.0 = 12.0']
[INFO] [1634841567.456886939] [calculator]: Timestamp of the message:
builtin_interfaces.msg.Time(sec=1634841567, nanosec=450401730)
[INFO] [1634841567.462630709] [calculator]: Subscribed argument a: 5.0
[INFO] [1634841567.467171573] [calculator]: Subscribed argument b: 5.0
[INFO] [1634841568.271546469] [calculator]: 5.0 / 5.0 = 1.0
[INFO] [1634841568.452373422] [calculator]: Timestamp of the message:
builtin_interfaces.msg.Time(sec=1634841568, nanosec=450399142)
[INFO] [1634841568.453181024] [calculator]: Subscribed argument a: 6.0
[INFO] [1634841568.454768417] [calculator]: Subscribed argument b: 3.0
[INFO] [1634841569.478251865] [calculator]: Timestamp of the message:
builtin_interfaces.msg.Time(sec=1634841569, nanosec=450370891)
[INFO] [1634841569.479135233] [calculator]: Subscribed argument a: 2.0
[INFO] [1634841569.486092150] [calculator]: Subscribed argument b: 4.0
[INFO] [1634841570.452583621] [calculator]: Timestamp of the message:
builtin_interfaces.msg.Time(sec=1634841570, nanosec=450576098)
[INFO] [1634841570.454423272] [calculator]: Subscribed argument a: 6.0
[INFO] [1634841570.455078341] [calculator]: Subscribed argument b: 8.0
[INFO] [1634841571.455307945] [calculator]: Timestamp of the message:
builtin_interfaces.msg.Time(sec=1634841571, nanosec=450346238)
[INFO] [1634841571.459106082] [calculator]: Subscribed argument a: 3.0
[INFO] [1634841571.462052332] [calculator]: Subscribed argument b: 5.0
```

jasmine@jasmine-VirtualBox: ~/robot_ws 34

argument

```
ument]: Published argument b: 9.0
[I                                                              rg
um                                                              0
[INFO] [1634841564.461568802] [arg
ument]: Published argument b: 1.0
[INFO] [1634841565.451848327] [arg
ument]: Published argument a: 9.0
[INFO] [1634841565.454375064] [arg
ument]: Published argument b: 3.0
[INFO] [1634841566.459632325] [arg
ument]: Published argument a: 3.0
[INFO] [1634841566.468041317] [arg
ument]: Published argument b: 6.0
[INFO] [1634841567.457564824] [arg
ument]: Published argument a: 5.0
[INFO] [1634841567.464109728] [arg
ument]: Published argument b: 5.0
[INFO] [1634841568.451116482] [arg
ument]: Published argument a: 6.0
[INFO] [1634841568.454882599] [arg
ument]: Published argument b: 3.0
[INFO] [1634841569.467685631] [arg
ument]: Published argument a: 2.0
[INFO] [1634841569.484415958] [arg
ument]: Published argument b: 4.0
[INFO] [1634841570.451311907] [arg
ument]: Published argument a: 6.0
[INFO] [1634841570.453834937] [arg
ument]: Published argument b: 8.0
[INFO] [1634841571.456704387] [arg
ument]: Published argument a: 3.0
[INFO] [1634841571.458250850] [arg
ument]: Published argument b: 5.0
```

jasmine@jasmine-VirtualBox: ~/robot_ws 48x34

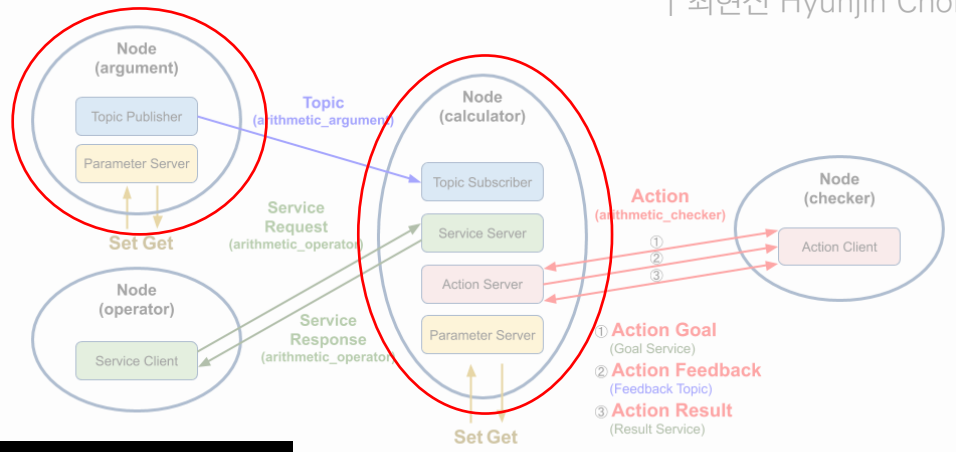
operator

```
Press Enter for next service call.
[INF                                                            esult
: 1
Press Enter for next service call.
[INFO] [1634841530.220218166] [operator]: Result
: 13.0
Press Enter for next service call.
[INFO] [1634841541.838736252] [operator]: Result
: -3.0
Press Enter for next service call.
[INFO] [1634841547.458621738] [operator]: Result
: 11.0
Press Enter for next service call.
[INFO] [1634841549.517361073] [operator]: Result
: -6.0
Press Enter for next service call.
[INFO] [1634841550.716128763] [operator]: Result
: 8.0
Press Enter for next service call.
[INFO] [1634841552.080283742] [operator]: Result
: 5.0
Press Enter for next service call.
[INFO] [1634841561.421703363] [operator]: Result
: 12.0
Press Enter for next service call.
[INFO] [1634841563.503221055] [operator]: Result
: 0.0
Press Enter for next service call.
[INFO] [1634841565.468586483] [operator]: Result
: 12.0
Press Enter for next service call.
[INFO] [1634841568.273017675] [operator]: Result
: 1.0
Press Enter for next service call.
```

```
jasmine@jasmine-VirtualBox:~/robot_ws$ ros2 run
topic_service_action_rclpy_example checker
[INFO] [1634841561.246101935] [checker]: Action
goal accepted.
[INFO] [1634841562.185097008] [checker]: Action
feedback: ['3.0 + 2.0 = 5.0', '6.0 * 2.0 = 12.0'
]
[INFO] [1634841563.199379741] [checker]: Action
feedback: ['3.0 + 2.0 = 5.0', '6.0 * 2.0 = 12.0'
, '6.0 * 2.0 = 12.0']
[INFO] [1634841564.201956946] [checker]: Action
feedback: ['3.0 + 2.0 = 5.0', '6.0 * 2.0 = 12.0'
, '6.0 * 2.0 = 12.0', '9.0 - 9.0 = 0.0']
[INFO] [1634841565.203754489] [checker]: Action
feedback: ['3.0 + 2.0 = 5.0', '6.0 * 2.0 = 12.0'
, '6.0 * 2.0 = 12.0', '9.0 - 9.0 = 0.0', '9.0 -
9.0 = 0.0']
[INFO] [1634841566.206588866] [checker]: Action
feedback: ['3.0 + 2.0 = 5.0', '6.0 * 2.0 = 12.0'
, '6.0 * 2.0 = 12.0', '9.0 - 9.0 = 0.0', '9.0 -
9.0 = 0.0', '9.0 + 3.0 = 12.0']
[INFO] [1634841567.216193225] [checker]: Action
feedback: ['3.0 + 2.0 = 5.0', '6.0 * 2.0 = 12.0'
, '6.0 * 2.0 = 12.0', '9.0 - 9.0 = 0.0', '9.0 -
9.0 = 0.0', '9.0 + 3.0 = 12.0', '9.0 + 3.0 = 12.
0']
[INFO] [1634841568.231340327] [checker]: Action
succeeded!
[INFO] [1634841568.234949290] [checker]: Action
result(all formula): ['3.0 + 2.0 = 5.0', '6.0 *
2.0 = 12.0', '6.0 * 2.0 = 12.0', '9.0 - 9.0 = 0.
0', '9.0 - 9.0 = 0.0', '9.0 + 3.0 = 12.0', '9.0
+ 3.0 = 12.0']
[INFO] [1634841568.238171353] [checker]: Action
result(total sum): 53.0
```

6.6. 실행

- 6.6.5 런치파일 실행
  - argument 노드와 calculator 노드를 한 번에 실행
  - (2부 18장에서 런치 프로그래밍)

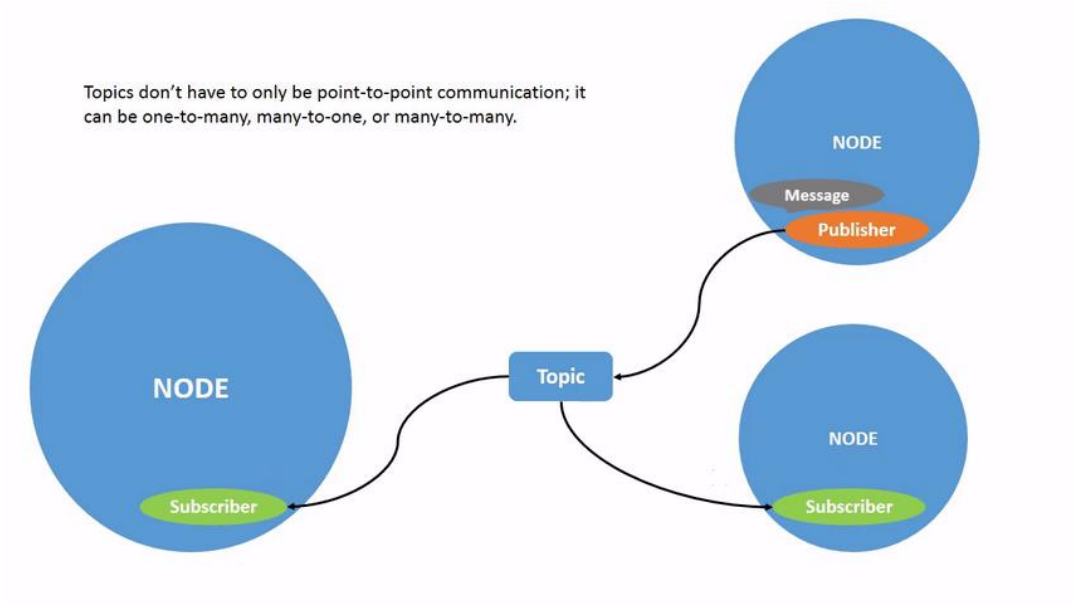


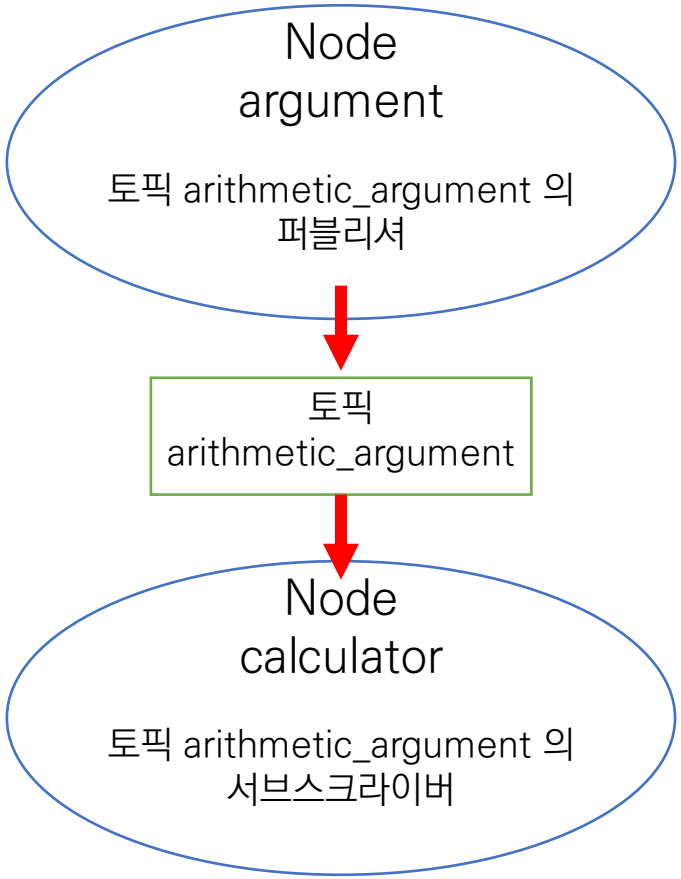

```
jasmine@jasmine-VirtualBox:~/robot_ws$ ros2 launch topic_service_action_rclpy_example arithmetic.launch.py
[INFO] [launch]: All log files can be found below /home/jasmine/.ros/log/2021-10-22-03-45-49-324571-jasmine-VirtualBox-6510
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [argument-1]: process started with pid [6512]
[INFO] [calculator-2]: process started with pid [6514]
[argument-1] [INFO] [1634841952.503173725] [argument]: Published argument a: 0.0
[argument-1] [INFO] [1634841952.506152782] [argument]: Published argument b: 4.0
[calculator-2] [INFO] [1634841952.531498030] [calculator]: Timestamp of the message: builtin_interfaces.msg.Time(sec=1634841952, nanosec=399934567)
[calculator-2] [INFO] [1634841952.534759915] [calculator]: Subscribed argument a: 0.0
[calculator-2] [INFO] [1634841952.535833074] [calculator]: Subscribed argument b: 4.0
[argument-1] [INFO] [1634841953.403073070] [argument]: Published argument a: 5.0
[calculator-2] [INFO] [1634841953.427806540] [calculator]: Timestamp of the message: builtin_interfaces.msg.Time(sec=1634841953, nanosec=399788788)
[argument-1] [INFO] [1634841953.432881193] [argument]: Published argument b: 9.0
[calculator-2] [INFO] [1634841953.435075614] [calculator]: Subscribed argument a: 5.0
[calculator-2] [INFO] [1634841953.439269860] [calculator]: Subscribed argument b: 9.0
[calculator-2] [INFO] [1634841953.454204942] [calculator]: Timestamp of the message: builtin_interfaces.msg.Time(sec=1634841953, nanosec=451011878)
[calculator-2] [INFO] [1634841953.455285238] [calculator]: Subscribed argument a: 9.0
[calculator-2] [INFO] [1634841953.459397347] [calculator]: Subscribed argument b: 2.0
[calculator-2] [INFO] [1634841954.422689328] [calculator]: Timestamp of the message: builtin_interfaces.msg.Time(sec=1634841954, nanosec=400133515)
[argument-1] [INFO] [1634841954.439883733] [argument]: Published argument a: 4.0
[argument-1] [INFO] [1634841954.443128397] [argument]: Published argument b: 3.0
[calculator-2] [INFO] [1634841954.464427898] [calculator]: Timestamp of the message: builtin_interfaces.msg.Time(sec=1634841954, nanosec=450607045)
[calculator-2] [INFO] [1634841954.467081589] [calculator]: Subscribed argument a: 4.0
[calculator-2] [INFO] [1634841954.468083822] [calculator]: Subscribed argument b: 2.0
[calculator-2] [INFO] [1634841954.469605500] [calculator]: Subscribed argument a: 8.0
[calculator-2] [INFO] [1634841954.470280139] [calculator]: Subscribed argument b: 2.0
[calculator-2] [INFO] [1634841955.407199807] [calculator]: Timestamp of the message: builtin_interfaces.msg.Time(sec=1634841955, nanosec=399271698)
[argument-1] [INFO] [1634841955.408679997] [argument]: Published argument a: 2.0
[argument-1] [INFO] [1634841955.410199306] [argument]: Published argument b: 1.0
[calculator-2] [INFO] [1634841955.420499926] [calculator]: Subscribed argument a: 2.0
[calculator-2] [INFO] [1634841955.423476649] [calculator]: Subscribed argument b: 1.0
[calculator-2] [INFO] [1634841955.461104391] [calculator]: Timestamp of the message: builtin_interfaces.msg.Time(sec=1634841955, nanosec=453743072)
```

7장 토픽 프로그래밍 (파이썬)

7.1 토픽

- 비동기식 단방향 메시지 송수신 방식
- 퍼블리셔(publisher)와 서브스크라이버(subscriber)간의 통신





7.2 토픽 퍼블리셔 코드 (argument node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/arithmetic/argument.py

```python
import random

from msg_srv_action_interface_example.msg import ArithmeticArgument
from rcl_interfaces.msg import SetParametersResult
import rclpy
from rclpy.node import Node
from rclpy.parameter import Parameter
from rclpy.qos import QoSDurabilityPolicy
from rclpy.qos import QoSHistoryPolicy
from rclpy.qos import QoSProfile
from rclpy.qos import QoSReliabilityPolicy


class Argument(Node):

    def __init__(self):
        super().__init__('argument')
        self.declare_parameter('qos_depth', 10)
        qos_depth = self.get_parameter('qos_depth').value
        self.declare_parameter('min_random_num', 0)
        self.min_random_num = self.get_parameter('min_random_num').value
        self.declare_parameter('max_random_num', 9)
        self.max_random_num = self.get_parameter('max_random_num').value
        self.add_on_set_parameters_callback(self.update_parameter)

        QOS_RKL10V = QoSProfile(
            reliability=QoSReliabilityPolicy.RELIABLE,
            history=QoSHistoryPolicy.KEEP_LAST,
            depth=qos_depth,
            durability=QoSDurabilityPolicy.VOLATILE)

        self.arithmetic_argument_publisher = self.create_publisher(
            ArithmeticArgument,
            'arithmetic_argument',
            QOS_RKL10V)

        self.timer = self.create_timer(1.0, self.publish_random_arithmetic_arguments)

    def publish_random_arithmetic_arguments(self):
        msg = ArithmeticArgument()
        msg.stamp = self.get_clock().now().to_msg()
        msg.argument_a = float(random.randint(self.min_random_num, self.max_random_num))
        msg.argument_b = float(random.randint(self.min_random_num, self.max_random_num))
        self.arithmetic_argument_publisher.publish(msg)
        self.get_logger().info('Published argument a: {0}'.format(msg.argument_a))
        self.get_logger().info('Published argument b: {0}'.format(msg.argument_b))

    def update_parameter(self, params):
        for param in params:
            if param.name == 'min_random_num' and param.type_ == Parameter.Type.INTEGER:
                self.min_random_num = param.value
            elif param.name == 'max_random_num' and param.type_ == Parameter.Type.INTEGER:
                self.max_random_num = param.value
        return SetParametersResult(successful=True)


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


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

```python
class Argument(Node):

    def __init__(self):
        super().__init__('argument')
        self.declare_parameter('qos_depth', 10)
```

Argument 클래스

- rclpy.node 모듈의 Node 클래스 상속
- 생성자를 통해 노드명 ‘argument’로 초기화

```python
QOS_RKL10V = QoSProfile(
    reliability=QoSReliabilityPolicy.RELIABLE,
    history=QoSHistoryPolicy.KEEP_LAST,
    depth=qos_depth,
    durability=QoSDurabilityPolicy.VOLATILE)
```

QoS 설정

7.2 토픽 퍼블리셔 코드 (argument node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/arithmetic/argument.py

```python
import random

from msg_srv_action_interface_example.msg import ArithmeticArgument
from rcl_interfaces.msg import SetParametersResult
import rclpy
from rclpy.node import Node
from rclpy.parameter import Parameter
from rclpy.qos import QoSDurabilityPolicy
from rclpy.qos import QoSHistoryPolicy
from rclpy.qos import QoSProfile
from rclpy.qos import QoSReliabilityPolicy


class Argument(Node):

    def __init__(self):
        super().__init__('argument')
        self.declare_parameter('qos_depth', 10)
        qos_depth = self.get_parameter('qos_depth').value
        self.declare_parameter('min_random_num', 0)
        self.min_random_num = self.get_parameter('min_random_num').value
        self.declare_parameter('max_random_num', 9)
        self.max_random_num = self.get_parameter('max_random_num').value
        self.add_on_set_parameters_callback(self.update_parameter)

        QOS_RKL10V = QoSProfile(
            reliability=QoSReliabilityPolicy.RELIABLE,
            history=QoSHistoryPolicy.KEEP_LAST,
            depth=qos_depth,
            durability=QoSDurabilityPolicy.VOLATILE)

        self.arithmetic_argument_publisher = self.create_publisher(
            ArithmeticArgument,
            'arithmetic_argument',
            QOS_RKL10V)

        self.timer = self.create_timer(1.0, self.publish_random_arithmetic_arguments)

    def publish_random_arithmetic_arguments(self):
        msg = ArithmeticArgument()
        msg.stamp = self.get_clock().now().to_msg()
        msg.argument_a = float(random.randint(self.min_random_num, self.max_random_num))
        msg.argument_b = float(random.randint(self.min_random_num, self.max_random_num))
        self.arithmetic_argument_publisher.publish(msg)
        self.get_logger().info('Published argument a: {0}'.format(msg.argument_a))
        self.get_logger().info('Published argument b: {0}'.format(msg.argument_b))

    def update_parameter(self, params):
        for param in params:
            if param.name == 'min_random_num' and param.type_ == Parameter.Type.INTEGER:
                self.min_random_num = param.value
            elif param.name == 'max_random_num' and param.type_ == Parameter.Type.INTEGER:
                self.max_random_num = param.value
        return SetParametersResult(successful=True)


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


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

퍼블리셔 설정

Node 클래스의 create_publisher 함수

```python
self.arithmetic_argument_publisher = self.create_publisher(
    ArithmeticArgument,      # 토픽 인터페이스 타입 (.msg)
    'arithmetic_argument',   # 토픽 이름
    QOS_RKL10V)              # QoS 설정
```

7.2 토픽 퍼블리셔 코드 (argument node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/arithmetic/argument.py

```python
import random

from msg_srv_action_interface_example.msg import ArithmeticArgument
from rcl_interfaces.msg import SetParametersResult
import rclpy
from rclpy.node import Node
from rclpy.parameter import Parameter
from rclpy.qos import QoSDurabilityPolicy
from rclpy.qos import QoSHistoryPolicy
from rclpy.qos import QoSProfile
from rclpy.qos import QoSReliabilityPolicy


class Argument(Node):

    def __init__(self):
        super().__init__('argument')
        self.declare_parameter('qos_depth', 10)
        qos_depth = self.get_parameter('qos_depth').value
        self.declare_parameter('min_random_num', 0)
        self.min_random_num = self.get_parameter('min_random_num').value
        self.declare_parameter('max_random_num', 9)
        self.max_random_num = self.get_parameter('max_random_num').value
        self.add_on_set_parameters_callback(self.update_parameter)

        QOS_RKL10V = QoSProfile(
            reliability=QoSReliabilityPolicy.RELIABLE,
            history=QoSHistoryPolicy.KEEP_LAST,
            depth=qos_depth,
            durability=QoSDurabilityPolicy.VOLATILE)

        self.arithmetic_argument_publisher = self.create_publisher(
            ArithmeticArgument,
            'arithmetic_argument',
            QOS_RKL10V)

        self.timer = self.create_timer(1.0, self.publish_random_arithmetic_arguments)

    def publish_random_arithmetic_arguments(self):
        msg = ArithmeticArgument()
        msg.stamp = self.get_clock().now().to_msg()
        msg.argument_a = float(random.randint(self.min_random_num, self.max_random_num))
        msg.argument_b = float(random.randint(self.min_random_num, self.max_random_num))
        self.arithmetic_argument_publisher.publish(msg)
        self.get_logger().info('Published argument a: {0}'.format(msg.argument_a))
        self.get_logger().info('Published argument b: {0}'.format(msg.argument_b))

    def update_parameter(self, params):
        for param in params:
            if param.name == 'min_random_num' and param.type_ == Parameter.Type.INTEGER:
                self.min_random_num = param.value
            elif param.name == 'max_random_num' and param.type_ == Parameter.Type.INTEGER:
                self.max_random_num = param.value
        return SetParametersResult(successful=True)


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


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

```python
self.timer = self.create_timer(1.0, self.publish_random_arithmetic_arguments)
```

1초마다 publish_random_arithmetic_arguments 함수 실행

ArithmeticArgument

```
# Messages
builtin_interfaces/Time stamp
float32 argument_a
float32 argument_b
```

publish_random_arithmetic_arguments 함수 정의

```python
def publish_random_arithmetic_arguments(self):
    msg = ArithmeticArgument()    msg 변수 생성. 인터페이스 클래스
    msg.stamp = self.get_clock().now().to_msg()
    msg.argument_a = float(random.randint(self.min_random_num, self.max_random_num))
    msg.argument_b = float(random.randint(self.min_random_num, self.max_random_num))
★   self.arithmetic_argument_publisher.publish(msg)
    self.get_logger().info('Published argument a: {0}'.format(msg.argument_a))
    self.get_logger().info('Published argument b: {0}'.format(msg.argument_b))
```

7.3 토픽 서브스크라이버 코드 (calculator node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/calculator/calculator.py

```python
from msg_srv_action_interface_example.action import ArithmeticChecker
from msg_srv_action_interface_example.msg import ArithmeticArgument
from msg_srv_action_interface_example.srv import ArithmeticOperator
from rclpy.action import ActionServer
from rclpy.callback_groups import ReentrantCallbackGroup
from rclpy.node import Node
from rclpy.qos import QoSDurabilityPolicy
from rclpy.qos import QoSHistoryPolicy
from rclpy.qos import QoSProfile
from rclpy.qos import QoSReliabilityPolicy


class Calculator(Node):

    def __init__(self):
        super().__init__('calculator')
        self.argument_a = 0.0
        self.argument_b = 0.0
        self.argument_operator = 0
        self.argument_result = 0.0
        self.argument_formula = ''
        self.operator = ['+', '-', '*', '/']
        self.callback_group = ReentrantCallbackGroup()

        self.declare_parameter('qos_depth', 10)
        qos_depth = self.get_parameter('qos_depth').value

        QOS_RKL10V = QoSProfile(
            reliability=QoSReliabilityPolicy.RELIABLE,
            history=QoSHistoryPolicy.KEEP_LAST,
            depth=qos_depth,
            durability=QoSDurabilityPolicy.VOLATILE)
```

Calculator 클래스
- Rclpy.node 모듈의 Node 클래스 상속
- 생성자를 통해 노드명 ‘calculator’로 초기화
- QoS 설정

7.3 토픽 서브스크라이버 코드 (calculator node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/calculator/calculator.py

서브스크라이버 설정

Node 클래스의 create_subscription 함수

```
self.arithmetic_argument_subscriber = self.create_subscription(
    ArithmeticArgument,                 토픽 인터페이스 타입 (.msg)
    'arithmetic_argument',              토픽 이름
    self.get_arithmetic_argument,       콜백함수. 메시지 서브스크라이브 할 때마다 실행
    QOS_RKL10V,                         QoS 설정
    callback_group=self.callback_group)
```

콜백함수

```
def get_arithmetic_argument(self, msg):
    self.argument_a = msg.argument_a
    self.argument_b = msg.argument_b
    self.get_logger().info('Timestamp of the message: {0}'.format(msg.stamp))
    self.get_logger().info('Subscribed argument a: {0}'.format(self.argument_a))
    self.get_logger().info('Subscribed argument b: {0}'.format(self.argument_b))
```

7.4 노드 실행 코드

- topic_service_action_rclpy_example/setup.py

```
entry_points={
    'console_scripts': [
        'argument = topic_service_action_rclpy_example.arithmetic.argument:main',
        'operator = topic_service_action_rclpy_example.arithmetic.operator:main',
        'calculator = topic_service_action_rclpy_example.calculator.main:main',
        'checker = topic_service_action_rclpy_example.checker.main:main',
    ],
},
```

/arithmetic/argument.py

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


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

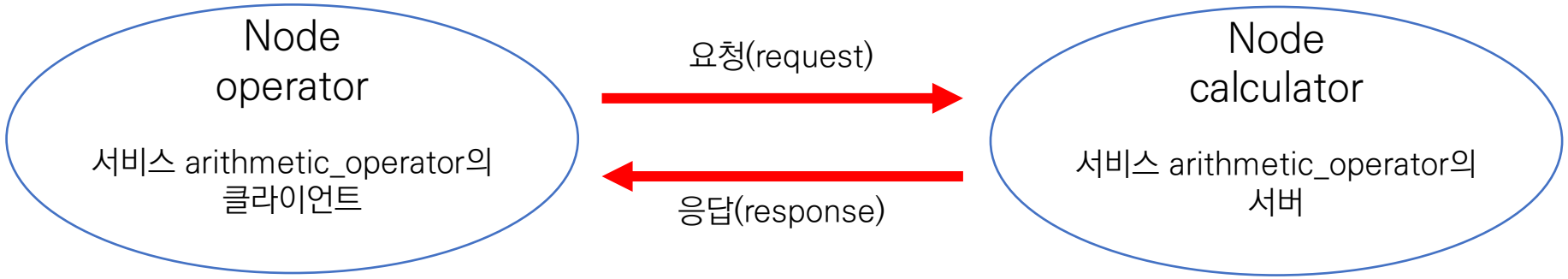
/calculator/main.py

```
import rclpy
from rclpy.executors import MultiThreadedExecutor

from topic_service_action_rclpy_example.calculator.calculator import Calculator


def main(args=None):
    rclpy.init(args=args)
    try:
        calculator = Calculator()
        executor = MultiThreadedExecutor(num_threads=4)
        executor.add_node(calculator)
        try:
            executor.spin()
        except KeyboardInterrupt:
            calculator.get_logger().info('Keyboard Interrupt (SIGINT)')
        finally:
            executor.shutdown()
            calculator.arithmetic_action_server.destroy()
            calculator.destroy_node()
    finally:
        rclpy.shutdown()


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

8장 서비스 프로그래밍 (파이썬)

8.1 서비스

- 동기식 양방향 메시지 송수신 방식
- 서비스 클라이언트(Service client)의 요청(Request)를 받아 서비스 서버(Service server)가 수행한 후 응답(Response)한다.



Interface:
ArithmeticOperator.srv

```
# Constants
int8 PLUS = 1
int8 MINUS = 2
int8 MULTIPLY = 3
int8 DIVISION = 4

# Request
int8 arithmetic_operator
---
# Response
float32 arithmetic_result
```

8.2 서비스 서버 코드 (calculator node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/calculator/calculator.py

서비스 서버

Node 클래스의 create_service 함수

```
self.arithmetic_service_server = self.create_service(
    ArithmeticOperator,          서비스 인터페이스 타입 (.srv)
    'arithmetic_operator',       서비스 이름
    self.get_arithmetic_operator,   Request 받으면 수행하는 콜백
    callback_group=self.callback_group)
```

```
def get_arithmetic_operator(self, request, response):
    self.argument_operator = request.arithmetic_operator
    self.argument_result = self.calculate_given_formula(
        self.argument_a,
        self.argument_b,        서브스크라이버가 저장해둔 멤버 변수
        self.argument_operator)
    response.arithmetic_result = self.argument_result
    self.argument_formula = '{0} {1} {2} = {3}'.format(
            self.argument_a,
            self.operator[self.argument_operator-1],
            self.argument_b,
            self.argument_result)
    self.get_logger().info(self.argument_formula)
    return response
```

Interface:
ArithmeticOperator.srv

```
# Constants
int8 PLUS = 1
int8 MINUS = 2
int8 MULTIPLY = 3
int8 DIVISION = 4

# Request
int8 arithmetic_operator
---
# Response
float32 arithmetic_result
```

8.2 서비스 서버 코드 (calculator node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/calculator/calculator.py

```
def calculate_given_formula(self, a, b, operator):
    if operator == ArithmeticOperator.Request.PLUS:
        self.argument_result = a + b
    elif operator == ArithmeticOperator.Request.MINUS:
        self.argument_result = a - b
    elif operator == ArithmeticOperator.Request.MULTIPLY:
        self.argument_result = a * b
    elif operator == ArithmeticOperator.Request.DIVISION:
        try:
            self.argument_result = a / b
        except ZeroDivisionError:
            self.get_logger().error('ZeroDivisionError!')
            self.argument_result = 0.0
            return self.argument_result
    else:
        self.get_logger().error(
            'Please make sure arithmetic operator(plus, minus, multiply, division).')
        self.argument_result = 0.0
    return self.argument_result
```

8.3 서비스 클라이언트 코드 (operator node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/arithmetic/operator.py

```python
import random

from msg_srv_action_interface_example.srv import ArithmeticOperator
import rclpy
from rclpy.node import Node


class Operator(Node):

    def __init__(self):
        super().__init__('operator')

        self.arithmetic_service_client = self.create_client(
            ArithmeticOperator,
            'arithmetic_operator')

        while not self.arithmetic_service_client.wait_for_service(timeout_sec=0.1):
            self.get_logger().warning('The arithmetic_operator service not available.')

    def send_request(self):
        service_request = ArithmeticOperator.Request()
        service_request.arithmetic_operator = random.randint(1, 4)
        futures = self.arithmetic_service_client.call_async(service_request)
        return futures


def main(args=None):
    rclpy.init(args=args)
    operator = Operator()
    future = operator.send_request()
    user_trigger = True
    try:
        while rclpy.ok():
            if user_trigger is True:
                rclpy.spin_once(operator)
                if future.done():
                    try:
                        service_response = future.result()
                    except Exception as e:  # noqa: B902
                        operator.get_logger().warn('Service call failed: {}'.format(str(e)))
                    else:
                        operator.get_logger().info(
                            'Result: {}'.format(service_response.arithmetic_result))
                        user_trigger = False
            else:
                input('Press Enter for next service call.')
                future = operator.send_request()
                user_trigger = True

    except KeyboardInterrupt:
        operator.get_logger().info('Keyboard Interrupt (SIGINT)')

    operator.destroy_node()
    rclpy.shutdown()


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

서비스 클라이언트

```python
class Operator(Node):

    def __init__(self):
        super().__init__('operator')
                                        # Node 클래스의 create_client 함수
        self.arithmetic_service_client = self.create_client(
            ArithmeticOperator,         # 서비스 인터페이스 타입 (.srv)
            'arithmetic_operator')      # 서비스 이름

        while not self.arithmetic_service_client.wait_for_service(timeout_sec=0.1):
            self.get_logger().warning('The arithmetic_operator service not available.')
```

```python
def send_request(self):
    service_request = ArithmeticOperator.Request()
    service_request.arithmetic_operator = random.randint(1, 4)
    futures = self.arithmetic_service_client.call_async(service_request)
    return futures
```

Interface:
ArithmeticOperator.srv

```
# Constants
int8 PLUS = 1
int8 MINUS = 2
int8 MULTIPLY = 3
int8 DIVISION = 4

# Request
int8 arithmetic_operator
---
# Response
float32 arithmetic_result
```

8.4 노드 실행 코드

- topic_service_action_rclpy_example/setup.py

```
entry_points={
    'console_scripts': [
        'argument = topic_service_action_rclpy_example.arithmetic.argument:main',
        'operator = topic_service_action_rclpy_example.arithmetic.operator:main',
        'calculator = topic_service_action_rclpy_example.calculator.main:main',
        'checker = topic_service_action_rclpy_example.checker.main:main',
    ],
},
```

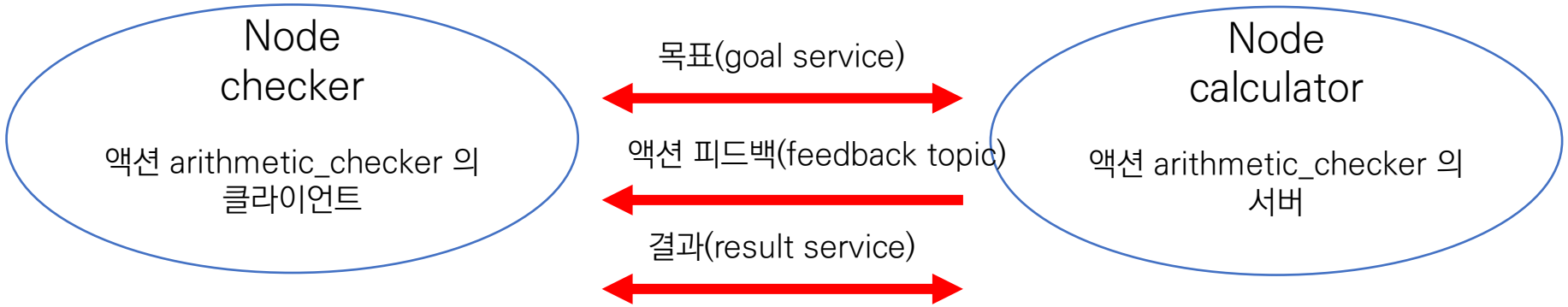
/arithmetic/operator.py

```
def main(args=None):
    rclpy.init(args=args)
    operator = Operator()   Operator 클래스를 operator 객체로 생성
    future = operator.send_request()
    user_trigger = True
    try:
        while rclpy.ok():
            if user_trigger is True:
                rclpy.spin_once(operator)
                if future.done():
                    try:
                        service_response = future.result()
                    except Exception as e:  # noqa: B902
                        operator.get_logger().warn('Service call failed: {}'.format(str(e)))
                    else:
                        operator.get_logger().info(
                            'Result: {}'.format(service_response.arithmetic_result))
                        user_trigger = False
            else:
                input('Press Enter for next service call.')
                future = operator.send_request()
                user_trigger = True

    except KeyboardInterrupt:
        operator.get_logger().info('Keyboard Interrupt (SIGINT)')

    operator.destroy_node()
    rclpy.shutdown()


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

9장 액션 프로그래밍 (파이썬)

9.1 액션

- 동기식+비동기식 양방향 메시지 송수신 방식
- 액션 클라이언트(Action client)의 목표(Goal)를 받아
  액션 서버(Action server)가 수행 중 중간 결과 피드백(Action feedback)과
  수행 완료 후 최종 결과(Action result)를 전송한다.



Interface:
ArithmeticChecker.action

```
# Goal
float32 goal_sum
---
# Result
string[] all_formula
float32 total_sum
---
# Feedback
string[] formula
```

9.2 액션 서버 코드 (calculator node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/calculator/calculator.py

액션 서버 rclpy.action 모듈의 ActionServer 클래스

```
self.arithmetic_action_server = ActionServer(
    self,
    ArithmeticChecker,    액션 인터페이스 타입 (.action)
    'arithmetic_checker',  액션 이름
    self.execute_checker,  goal 받으면 수행하는 콜백
    callback_group=self.callback_group)
```

```
def execute_checker(self, goal_handle):
    self.get_logger().info('Execute arithmetic_checker action!')
    feedback_msg = ArithmeticChecker.Feedback()
    feedback_msg.formula = []
    total_sum = 0.0
    goal_sum = goal_handle.request.goal_sum
    while total_sum < goal_sum:
        total_sum += self.argument_result
        feedback_msg.formula.append(self.argument_formula)
        self.get_logger().info('Feedback: {0}'.format(feedback_msg.formula))
        goal_handle.publish_feedback(feedback_msg)
        time.sleep(1)
    goal_handle.succeed()
    result = ArithmeticChecker.Result()
    result.all_formula = feedback_msg.formula
    result.total_sum = total_sum
    return result
```

Interface: ArithmeticChecker.action

```
# Goal
float32 goal_sum
---
# Result
string[] all_formula
float32 total_sum
---
# Feedback
string[] formula
```

9.3 액션 클라이언트 코드 (checker node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/checker/checker.py

액션 서버

```
class Checker(Node):

    def __init__(self):
        super().__init__('checker')
        self.arithmetic_action_client = ActionClient(
          self,
          ArithmeticChecker,
          'arithmetic_checker')
```

rclpy.action 모듈의 ActionServer 클래스

액션 인터페이스 타입 (.action)

액션 이름

```
def send_goal_total_sum(self, goal_sum):
    wait_count = 1
    while not self.arithmetic_action_client.wait_for_server(timeout_sec=0.1):
        if wait_count > 3:
            self.get_logger().warning('Arithmetic action server is not available.')
            return False
        wait_count += 1
    goal_msg = ArithmeticChecker.Goal()
    goal_msg.goal_sum = (float)(goal_sum)
    self.send_goal_future = self.arithmetic_action_client.send_goal_async(
        goal_msg,
        feedback_callback=self.get_arithmetic_action_feedback)
    self.send_goal_future.add_done_callback(self.get_arithmetic_action_goal)
    return True
```

Interface:
ArithmeticChecker.action

```
# Goal
float32 goal_sum
---
# Result
string[] all_formula
float32 total_sum
---
# Feedback
string[] formula
```

9.3 액션 클라이언트 코드 (checker node)

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/checker/checker.py

```python
def get_arithmetic_action_feedback(self, feedback_msg):
    action_feedback = feedback_msg.feedback.formula
    self.get_logger().info('Action feedback: {0}'.format(action_feedback))

def get_arithmetic_action_result(self, future):
    action_status = future.result().status
    action_result = future.result().result
    if action_status == GoalStatus.STATUS_SUCCEEDED:
        self.get_logger().info('Action succeeded!')
        self.get_logger().info(
            'Action result(all formula): {0}'.format(action_result.all_formula))
        self.get_logger().info(
            'Action result(total sum): {0}'.format(action_result.total_sum))
    else:
        self.get_logger().warning(
            'Action failed with status: {0}'.format(action_status))
```

Interface:
ArithmeticChecker.action

```
# Goal
float32 goal_sum
---
# Result
string[] all_formula
float32 total_sum
---
# Feedback
string[] formula
```

9.4 노드 실행 코드

- topic_service_action_rclpy_example/setup.py

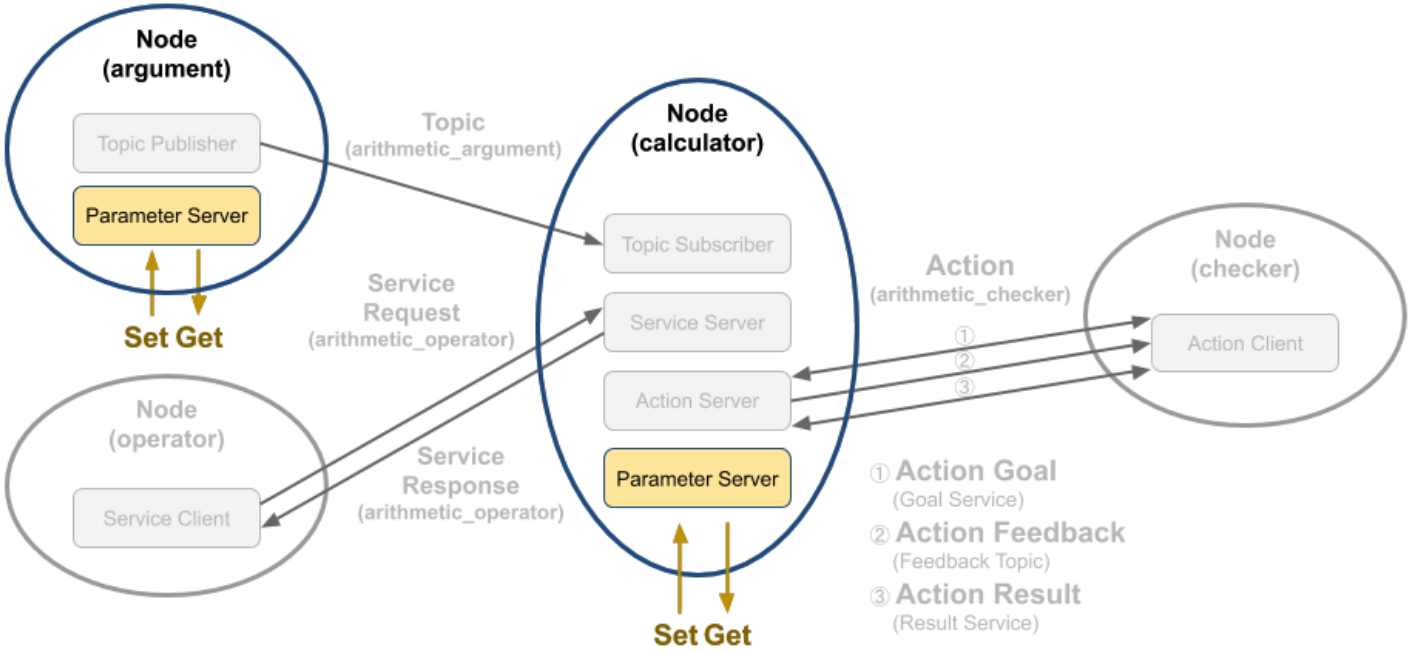
```
entry_points={
    'console_scripts': [
        'argument = topic_service_action_rclpy_example.arithmetic.argument:main',
        'operator = topic_service_action_rclpy_example.arithmetic.operator:main',
        'calculator = topic_service_action_rclpy_example.calculator.main:main',
        'checker = topic_service_action_rclpy_example.checker.main:main',
    ],
},
```

/checker/main.py

```
from topic_service_action_rclpy_example.checker.checker import Checker


def main(argv=sys.argv[1:]):
    parser = argparse.ArgumentParser(formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument(
        '-g',
        '--goal_total_sum',
        type=int,
        default=50,
        help='Target goal value of total sum')
    parser.add_argument(
        'argv', nargs=argparse.REMAINDER,
        help='Pass arbitrary arguments to the executable')
    args = parser.parse_args()

    rclpy.init(args=args.argv)
    try:
        checker = Checker()
        checker.send_goal_total_sum(args.goal_total_sum)
        try:
            rclpy.spin(checker)
        except KeyboardInterrupt:
            checker.get_logger().info('Keyboard Interrupt (SIGINT)')
        finally:
            checker.arithmetic_action_client.destroy()
            checker.destroy_node()
    finally:
        rclpy.shutdown()


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

10장 파라미터 프로그래밍(파이썬)

10.1 파라미터


Node
(argument)
Topic Publisher
Parameter Server
Set Get
Topic
(arithmetic_argument)
Node
(calculator)
Topic Subscriber
Service Server
Action Server
Parameter Server
Set Get
Service
Request
(arithmetic_operator)
Service
Response
(arithmetic_operator)
Node
(operator)
Service Client
Action
(arithmetic_checker)
① ② ③
Node
(checker)
Action Client
① Action Goal
(Goal Service)
② Action Feedback
(Feedback Topic)
③ Action Result
(Result Service)

10.2 파라미터 설정

- declare_parameter 함수
  - 노드에서 사용할 파라미터의 고유이름을 지정하고 초깃값 설정
- get_parameter 함수
  - 노드에서 사용할 파라미터 값을 불러오는 것. 주로 .yaml 파라미터 파일의 값을 불러옴
- add_on_set_parameters_callack
  - 서비스 형태로 파라미터 변경 요청이 있을 때 사용

10.2 파라미터 설정

- topic_service_action_rclpy_example/topic_service_action_rclpy_example/arithmetic/argument.py

```
self.declare_parameter('max_random_num', 9)
self.max_random_num = self.get_parameter('max_random_num').value
self.add_on_set_parameters_callback(self.update_parameter)
```

```
def update_parameter(self, params):
    for param in params:
        if param.name == 'min_random_num' and param.type_ == Parameter.Type.INTEGER:
            self.min_random_num = param.value
        elif param.name == 'max_random_num' and param.type_ == Parameter.Type.INTEGER:
            self.max_random_num = param.value
    return SetParametersResult(successful=True)
```

10.3. 파라미터 사용 방법 (CLI)

- argument 노드 실행 중

```
jasmine@jasmine-VirtualBox:~/robot_ws$ ros2 param list
/argument:
  max_random_num
  min_random_num
  qos_depth
```

- 파라미터 값 확인

```
jasmine@jasmine-VirtualBox:~/robot_ws$ ros2 param get /argument max_random_num
Integer value is: 9
```

- 파라미터 변경

```
jasmine@jasmine-VirtualBox:~/robot_ws$ ros2 param set /argument max_random_num 100
Set parameter successful
```

```
[INFO] [1634850563.185247796] [argument]: Published argument a: 8.0
[INFO] [1634850563.188531059] [argument]: Published argument b: 6.0
[INFO] [1634850564.180080319] [argument]: Published argument a: 0.0
[INFO] [1634850564.180783884] [argument]: Published argument b: 4.0
[INFO] [1634850565.181408556] [argument]: Published argument a: 1.0
[INFO] [1634850565.185282594] [argument]: Published argument b: 6.0
[INFO] [1634850566.181926519] [argument]: Published argument a: 13.0
[INFO] [1634850566.185374259] [argument]: Published argument b: 18.0
[INFO] [1634850567.190841221] [argument]: Published argument a: 11.0
[INFO] [1634850567.196728575] [argument]: Published argument b: 52.0
[INFO] [1634850568.180001872] [argument]: Published argument a: 13.0
[INFO] [1634850568.180773279] [argument]: Published argument b: 19.0
[INFO] [1634850569.180395901] [argument]: Published argument a: 30.0
[INFO] [1634850569.181366747] [argument]: Published argument b: 89.0
[INFO] [1634850570.179967970] [argument]: Published argument a: 51.0
[INFO] [1634850570.180682839] [argument]: Published argument b: 30.0
[INFO] [1634850571.181439571] [argument]: Published argument a: 71.0
[INFO] [1634850571.182737416] [argument]: Published argument b: 59.0
[INFO] [1634850572.185050287] [argument]: Published argument a: 95.0
[INFO] [1634850572.188932426] [argument]: Published argument b: 94.0
```

10.5 기본 파라미터 설정 방법

- param/arithmetic_cofig.yaml

```yaml
/**: # namespace and node name
  ros__parameters:
    qos_depth: 30
    min_random_num: 0
    max_random_num: 9
```

- launch/arithmetic.launch.py

```python
def generate_launch_description():
    param_dir = LaunchConfiguration(
        'param_dir',
        default=os.path.join(
            get_package_share_directory('topic_service_action_rclpy_example'),
            'param',
            'arithmetic_config.yaml'))
```

11장 실행인자 프로그래밍(파이썬)

11.1 실행 인자

- 프로그램 실행 시 옵션으로 추가하여 실행하는 인자
- Parameter 는 매개변수, Argument 는 실행 인자
- main 함수에서 매개변수를 통해 접근할 수 있다.

```
~/robot_ws$ ros2 run topic_service_action_rclpy_example checker -g 100
```

11.2 ROS2 에서의 실행 인자 처리

- 인수들을 무시할 때

```
def main(args=None):
    rclpy.init(args=args)
    operator = Operator()
```

11.3 실행 인자의 구문 해석

checker/main.py

```
def main(argv=sys.argv[1:]):
    parser = argparse.ArgumentParser(formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument(
        '-g',
        '--goal_total_sum',
        type=int,
        default=50,
```

11.3 실행 인자의 구문 해석

```python
def main(argv=sys.argv[1:]):
    parser = argparse.ArgumentParser(formatter_class=argparse.ArgumentDefaultsHelpFormatter)
    parser.add_argument(
        '-g',
        '--goal_total_sum',
        type=int,
        default=50,
        help='Target goal value of total sum')
    parser.add_argument(
        'argv', nargs=argparse.REMAINDER,
        help='Pass arbitrary arguments to the executable')
    args = parser.parse_args()

    rclpy.init(args=args.argv)
    try:
        checker = Checker()
        checker.send_goal_total_sum(args.goal_total_sum)
        try:
            rclpy.spin(checker)
        except KeyboardInterrupt:
            checker.get_logger().info('Keyboard Interrupt (SIGINT)')
        finally:
            checker.arithmetic_action_client.destroy()
            checker.destroy_node()
    finally:
        rclpy.shutdown()


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

파서 만들기 (argparse 모듈의 기본 형식)

인자 추가하기

인자 이름

인자 파싱하기

인자 사용하기

Thank you !