ROS 2 스터디 10주차

구본우

1장 Logging

- 1.1 로그(Log)

- 프로그램을 개발하는 과정에서 개발자가 프로그램을 검토하는 용도로 사용되고, 간단한 디버깅 툴로 활용된다.
- ROS 2에서는 logger와 logging(rclcpp, rclpy) 라이브러리를 사용해 로그를 남긴다.

1. 매우 단순한 인터페이스
2. 초기화 없이 사용 가능
3. 다양한 로그 수준 설정 가능
4. 다양한 필터링 기능 제공
5. Printf와 stream 스타일 제공
6. 런타임 성능에 최소한의 영향
7. 스레드 세이프
8. 출력되는 문구에 대한 자세한 정보 제공
9. 계층 구조
10. Launch 파일에서 로그 수준 설정 가능
11. 문서 저장 기능 제공
12. 런타임에서 로그 수준 변경 가능

1장 Logging

- 1.2 로그 설정
- 1.2.1 로그 경로 ($ ls ~/.ros/log)

- Galactic 버전부터 저장 경로 변경 가능

- 1.2.2 로그 수준

1) Programmatically: 가장 기본적인 방법으로 코드에 명시하는 것
2) Externally: 특정 노드의 로그 레벨을 런타임에서 변경 가능
3) Command line: 노드를 실행할 때 인자를 통해 해당 노드의 로그 수준 지정 가능

1장 Logging

- 1.2.3 로그 형식 설정
- 터미널 창에서 확인할 수 있는 로그 형식을 변경 가능

- 1.2.4 로그 색상 설정
- INFO(하얀색), WARN(노란색), ERROR(빨간색)

- 1.2.5 로그 스트림 설정
- 이전에는 스트림 버퍼가 모두 차지 않으면 DEBUG와 INFO 수준 로그가 정확한 타이밍에 표시되지 못함
- Foxy 이후에는 상관없음

- 1.2.6 로그 라인 버퍼링 설정
- 버퍼링을 사용하고 싶다면 환경변수를 통해 설정 가능

1장 Logging

- 1.2.3 로그 형식 설정
- 터미널 창에서 확인할 수 있는 로그 형식을 변경 가능

- 1.2.4 로그 색상 설정
- INFO(하얀색), WARN(노란색), ERROR(빨간색)

- 1.2.5 로그 스트림 설정
- 이전에는 스트림 버퍼가 모두 차지 않으면 DEBUG와 INFO 수준 로그가 정확한 타이밍에 표시되지 못함
- Foxy 이후에는 상관없음

- 1.2.6 로그 라인 버퍼링 설정
- 버퍼링을 사용하고 싶다면 환경변수를 통해 설정 가능

2장 ROS 2 CLI

- 2.1 ROS 2 CLI
- $ ros2 [verbs] [sub-verbs] [options] [arguments]

- 2.2 ROS 2 CLI 실습

- 2.3 ROS 2 CLI의 빠른 실행
- alias 단축 명령어를 지정하여 사용

- 2.4 CLI 명령어에서 ROS arguments 사용하기
- -r __ns:=사용할 네임스페이스
- -r __node:=변경할 노드 이름
- -r 본래의 토픽/서비스/액션명:=변경할 이름
- -p 파라미터 이름:=변경할 파라미터값
- --params-file 파라미터 파일

2장 ROS 2 CLI

- 2.5 신규 ROS 2 CLI 작성 방법
- 2.5.1 실행 예제

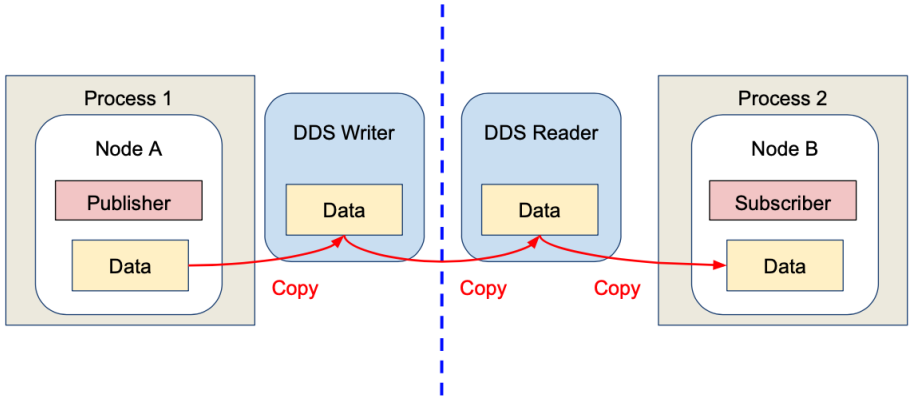
- ros2-seminar-examples 리포지터리를 내려받아 빌드하였다면 ros2 명령어에 env 포함됨
- Ros2 env를 입력하면 도움말과 verb로 사용할 수 있는 list와 set을 확인 가능

- 2.5.2 소스코드

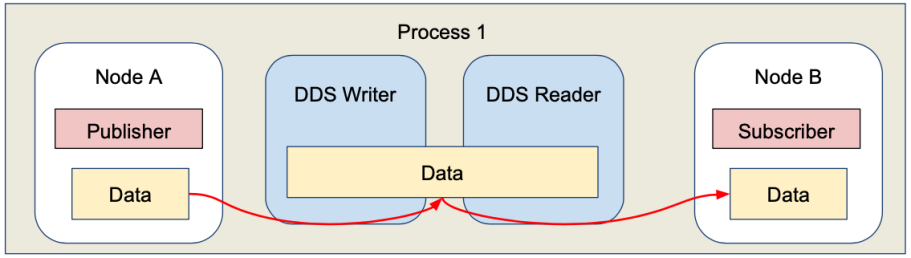
- ros2env 패키지는 다른 ros2cli와 동일한 형태로 제작되었으며 일반적인 ROS 2 패키지와 크게 다르지 않다.
- 이는 파이썬 패키지로 만들어져 있으며 실행에 대한 정보는 패키지 설정 파일인 setup.py 파일 내의 entry_points를 보면 된다.
- 해당 패키지는 ros2cli.command를 이용하여 터미널 창에서 ros2 env로 실행된다.
- 그리고 ros2env.verb에 포함된 list와 set verbs를 사용할 수 있어서 ros2 env list 및 ros2 env set과 같이 사용하면 된다.

3장 Intra-process communication

ROS 2에서 퍼블리쉬 노드를 실행시키면 하나의 프로세스가 생성되고, 서브스크라이브 노드를 실행시키면 또 다른 프로세스가 생성된다.



하지만 ROS 2의 IPC를 사용하면 복수개의 노드를 단일 프로세스에서 동작되도록 할 수 있고, 고정된 메모리 공간에 접근하여 메모리 복사가 이루어지지 않도록 한다.



4장 QoS

- 4.1 QoS(Quality of Service)

- 옵션 6가지 (History, Reliability, Durability, Deadline, Lifespan, Liveliness)
- RMW QoS Profile

	Default	Sensor Data	Service	Action Status	Parameters	Parameter Events
Reliability	RELIABLE	BEST_EFFORT	RELIABLE	RELIABLE	RELIABLE	RELIABLE
History	KEEP_LAST	KEEP_LAST	KEEP_LAST	KEEP_LAST	KEEP_LAST	KEEP_LAST
Depth (History Depth)	10	5	10	1	1,000	1,000
Durability	VOLATILE	VOLATILE	VOLATILE	TRANSIENT LOCAL	VOLATILE	VOLATILE

4장 QoS

History	데이터 전송 시점 이후의 데이터를 몇개나 보관하냐를 결정하는 QoS 옵션
KEEP_LAST	정해진 메시지 큐 사이즈 만큼의 데이터를 보관 * depth: 메시지 큐의 사이즈 (KEEP_LAST 설정일 경우에만 유효)
KEEP_ALL	모든 데이터를 보관 (메시지 큐의 사이즈는 DDS 벤더마다 다름)

Durability	데이터를 수신하는 서브스크라이버가 생성되기 전의 데이터를 사용할지 폐기할지에 대한 QoS 옵션
TRANSIENT_LOCAL	Subscription이 생성되기 전의 데이터도 보관 (Publisher에만 적용 가능)
VOLATILE	Subscription이 생성되기 전의 데이터는 무효

Reliability	데이터 전송에 있어서 전송 속도에 최우선 정책을 펼치느냐 신뢰성에 최우선 정책을 펼치냐를 결정함
BEST_EFFORT	데이터 송신에 집중. 전송 속도를 중시하며 네트워크 상태에 따라 유실이 발생할 수 있음
RELIABLE	데이터 수신에 집중. 신뢰성을 중시하며 유실이 발생하면 재전송을 통해 수신을 보장함

Liveliness	정해진 주기 안에서 노드 혹은 토픽의 생사 확인하는 QoS 옵션
liveliness	자동 또는 매뉴얼로 확인할지를 지정하는 옵션, 하기 3가지 중 선택 * AUTOMATIC, MANUAL_BY_NODE, MANUAL_BY_TOPIC 중 선택
lease_duration	Liveliness을 확인하는 주기

Lifespan	정해진 주기 안에서 수신되는 데이터만 유효 판정하고 그렇지 않은 데이터는 삭제하는 QoS 옵션
lifespan_duration	Lifespan을 확인하는 주기

Deadline	정해진 주기 안에 데이터가 발신 및 수신되지 않을 경우 EventCallback를 실행시키는 QoS 옵션
deadline_duration	Deadline을 확인하는 주기

4장 QoS

- 4.2 Topic, Service, Action의 QoS 설정
- 4.2.1 Topic

- Topic의 기본 QoS 설정은 rclpy와 rclcpp에서 특별한 설정이 없는 한 RMW QoS Profile의 Default 설정을 사용함

- Reliability: RELIABLE
- History: KEEP_LAST
- Depth: 10
- Durability: VOLATILE

4장 QoS

- 4.2.2 Service

- Service의 기본 QoS 설정은 특별한 경우를 제외하고는 기본 QoS를 사용

- rclpy의 경우 rclpy.node 모듈의 Node 클래스의 서비스 서버를 설정하는 create_service 함수에서 qos_profile을 설정하지 않는 한 qos_profile_service_default가 기본으로 사용된다.

- rclcpp 또한 rclcpp/node.hpp의 create_service 함수에서 rmw_qos_profile_services_defaul가 기본으로 설정되어 있다.

4장 QoS

- 4.2.3 Action

- 액션은 토픽과 서비스를 모두 사용하는 복합 형태

- QoS 또한 서비스 계열과 토픽 계열을 각각 설정하도록 되어 있지만 서비스는 qos_profile_services_defaul를 기본으로 설정하여 사용한다.

- 피드백 퍼블리셔는 QoSProfile과 rmw_qos_profile_default를 초깃값으로 사용한다.

- 액션 상태 퍼블리셔는 액션만을 위해 나중에 추가된 프로파일인 qos_profile_action_status_default를 각각의 기본 설정값을 이용하게 된다.