ROS 로봇프로그래밍

-ROS1 완독하기 챌린지-

김예나

Ch4. ROS의 중요 콘셉트

4.1 ROS 용어 정리

ROS

- 로봇의 응용프로그램을 개발하기 위한 운영체제와 같은 로봇 소프트웨어 플랫폼
- 하드웨어 추상화, 하위 디바이스 제어, 센싱,인식, 지도 작성, 모션 플래닝 등의 기능 구현, 프로세스 간의 메시지 파싱, 패키징 관리, 개발환경에 필요한 라이브러리와 다양한 개발 및 디버깅 도구 제공

마스터

- 노드와 노드 사이 연결, 메시지 통신을 위한 네임 서버 역할
- roscore로 실행
- 슬레이브들과 접속상태를 유지하지 않는 HTTP 기반의 프로토콜 XMLRPC(XML-Remote Procedure Call)통신
  → 슬레이브인 노드들이 필요할 때만 접속하여 정보 등록 및 다른 노드의 정보 요청
  → 매우 가볍고, 다양한 프로그래밍 언어 지원
- ROS_MASTER_URI변수(기본값 URI주소: 현재로컬IP, 포트: 11311)에 주소 및 포트 설정 가능

4.1 ROS 용어 정리

노드

- ROS에서 실행되는 최소 단위의 프로세서, 하나의 실행 가능한 프로그램
- 마스터에 노드와 퍼블리셔, 서브스크라이버, 서비스 서버, 서비스 클라이언트 역할을 하는 노드, 토픽, 서비스의 각 이름, 메시지 형태, URI 주소와 포트 등록
  → 노드끼리 토픽과 서비스를 이용하여 메시지 주고받기 가능
- 마스터와 통신, 노드 간 접속요청 및 응답: XMLRPC,
  노드 간 메시지 통신: TCPROS이용(TCP/IP통신 계열)
- URI주소: 현재 노드가 실행 중인 컴퓨터에 저장된 ROS_HOSTNAME 환경 변수값,
  포트: 임의의 고유값

패키지

- ROS를 구성하는 기본 단위
- 최소 하나 이상의 노드 + 다른 패키지의 노드를 실행하기 위한 설정 파일
  + 각종 프로세스를 구동하기 위한 ROS 의존성 라이브러리, 데이터셋, 설정 파일 등 포함

메타패키지 – 공통된 목적을 지닌 패키지들의 집합

4.1 ROS 용어 정리

메시지

- 노드 간 데이터를 주고 받을 때 사용, integer,floating point, boolean과 같은 변수 형태
- 메시지 안에 메시지를 품고 있는 간단한 데이터 구조 or 배열 같은 구조도 사용 가능
- 통신 방법: TCPROS, UDPROS 방식 등
- 단방향 메시지 송수신 방식: 토픽, 양방향 메시지 요청/응답 방식: 서비스 이용

토픽

- 구성: **퍼블리셔** 노드 –(토픽등록)→ 마스터 –(퍼블리셔 정보)→ **서브스크라이버** 노드
- **비동기 방식**으로, 필요에 따라 주어진 데이터 전송하고 받기에 적합
- 한 번의 접속으로 **지속적**인 메시지 송수신 → 센서 데이터에 적합

서비스

- 구성: **서비스 서버** –(요청 입력 받은 후 서비스 수행 후 결과 출력)→
  ←——(요청을 출력하고, 응답을 입력받음) —— **서비스 클라이언트**
- **동기** 방식, 요청과 응답 **양방향** 통신
- 서비스의 요청과 응답이 완료되면 연결된 두 노드는 **끊어짐**

4.1 ROS 용어 정리

액션

- 서비스처럼 양방향을 요구 but 응답까지 오래 걸리고 중간 결과값이 필요할 때 사용
- 비동기식 양방향 메세지 통신
- 요청(목표), 응답(결과) + 중간결과값(피드백)
- 구성: **액션 서버** –(요청 입력 서비스 수행 후 결과 or 피드백 값 출력)→
  ←——(요청 출력, 결과 or 피드백 값을 입력) —— **액션 클라이언트**

파라미터

- 디폴트로 설정값이 지정되어 있고, 필요에 따라 외부에서 읽거나 쓰기 가능
- 외부에서 쓰기 기능으로 설정값을 실시간으로 변경 가능
- 외부 장치와 연결되는 PC의 USB포트나 카메라 캘리브레이션 값, 모터 속도나 명령어들의 최대/최소값 설정에 유용
- 마스터의 파라미터 서버에 등록

4.1 ROS 용어 정리

캐킨
- ROS의 빌드 시스템(rosbuild(X) → catkin(O) 사용하기!)
- CMake(Cross Platform Make)이용, 패키지 폴더 CMakeLists.txt파일에 빌드환경 기술

roscore : ros마스터 구동, 같은 네트워크에서 하나만 구동
rosrun: ros기본 실행 명령어, 패키지에서 하나의 노드 실행
roslaunch: 여러 노드 실행
+패키지의 파라미터나 노드 이름 변경, 노드 네임스페이스 설정, ROS_ROOT 및 ROS_PACKAGE_PATH 설정, 환경변수 변경 등에 사용

bag
- 주고 받은 메시지의 데이터 저장할 때 사용되는 파일 포맷, *.bag 사용
- 메시지 저장하고 필요할 때 재생하여 이전 상황을 그대로 재현 가능
  → 같은 실험 수행하지 않아도 당시 센서값을 반복해서 사용 가능

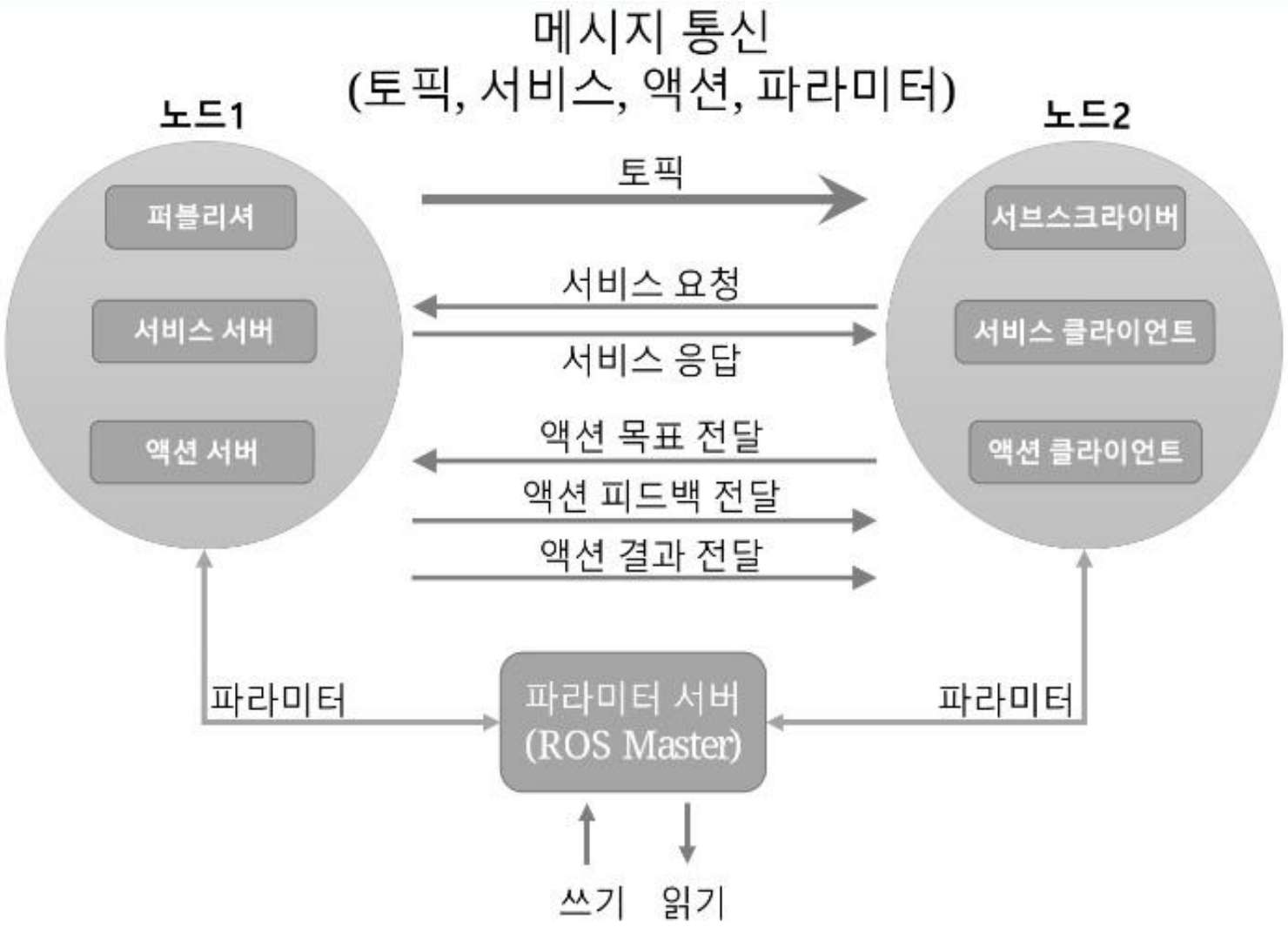
4.1 ROS 용어 정리

그래프
- 현재 실행 중인 메시지 통신을 그래프로 나타낸 것, 일회성의 서비스는 불가
- 명령어: rqt_graph 또는 rosrun rqt_graph rqt_graph

네임
- 노드, 파라미터, 토픽, 서비스 모두 네임이 있어 마스터에 등록 후 각 노드의 파라미터, 토픽, 서비스 사용시 이름을 기반으로 검색, 메시지 전송함
- 실행할 때 변경 가능, 같은 노드, 파라미터, 토픽, 서비스여도 다른 네임으로 중복 사용가능

bag
- 주고 받은 메시지의 데이터 저장할 때 사용되는 파일 포맷, *.bag 사용
- 메시지 저장하고 필요할 때 재생하여 이전 상황을 그대로 재현 가능
  → 같은 실험 수행하지 않아도 당시 센서값을 반복해서 사용 가능

4.2 메시지 통신



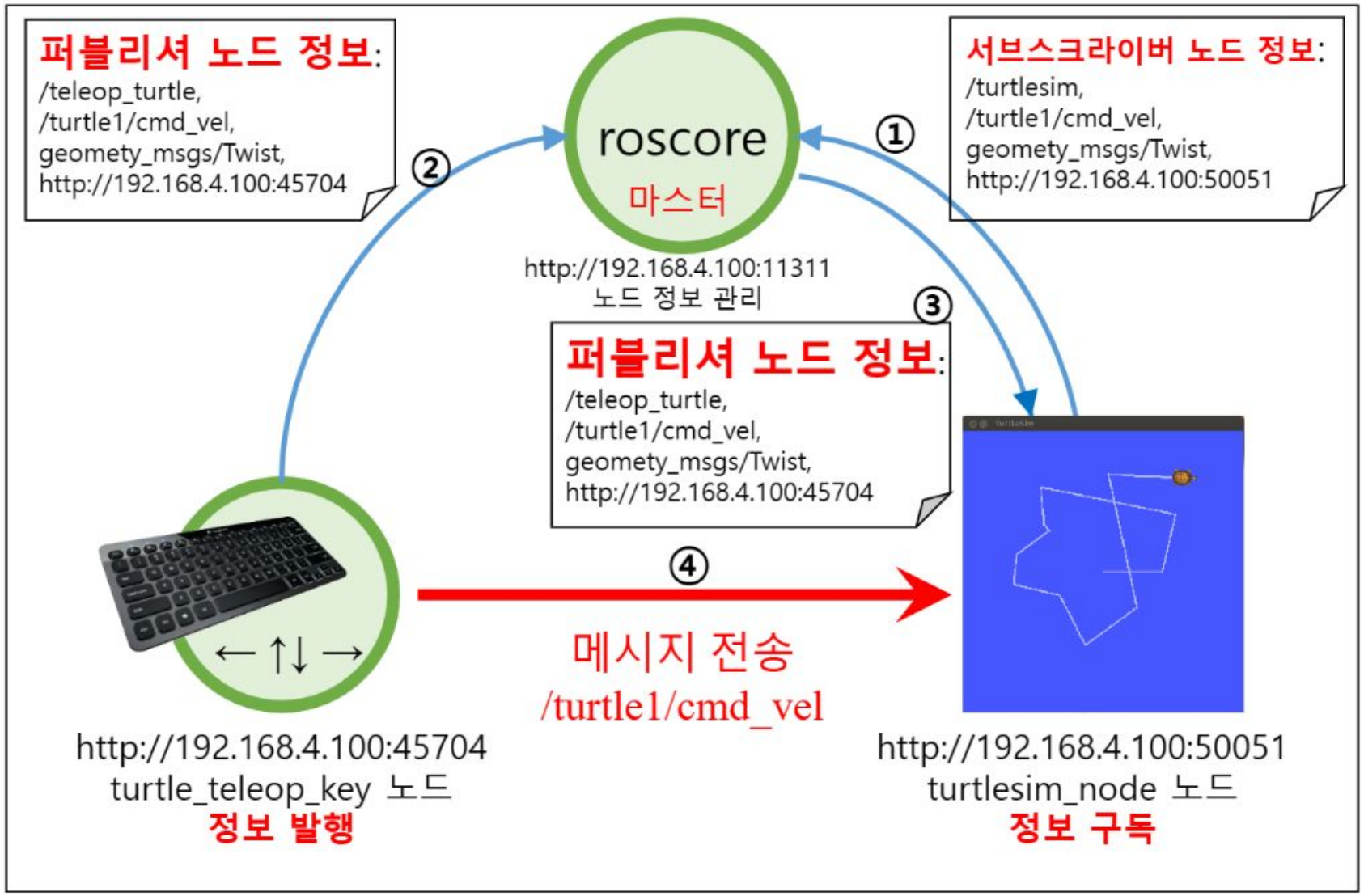
1) **토픽**: 비동기 단방향 연속성
   - N:N통신 가능
   - 센서 데이터에 적합

2) **서비스**: 동기 양방향 일회성
   - 로봇에 특정 동작 수행하도록 요청 or 특정 조건에 따라 이벤트 발생해야 할 노드에 사용

3) **액션**: 비동기 양방향 + 피드백
   - 임의의 시점에 목표취소 가능
   복잡한 로봇 업무 지시에 사용

4) **파라미터**: 외부 변경 가능

4.2 메시지 통신 – 통신 흐름(예제 turtlesim)



1) 마스터 구동

: 노드들의 이름, 토픽, 서비스 , 액션의 이름, 메시지 형태, URI주소와 포트 등록 받음

- ①, ② 퍼블리셔, 서브스크라이버 노드 정보 마스터에 등록 (XMLRPC)

2) TCPROS 접속 요청 및 응답

서브스크라이버 노드: 마스터로부터 받은 퍼블리셔 정보를 기반으로 직접 접속 요청

- ③

퍼블리셔 노드: 접속 응답에 자신의 URI주소와 포트전송

3) TCPROS 접속 및 통신 – ④

4.3 메시지

- 노드 간에 데이터를 주고받을 때 사용하는 데이터 형태
- 정수(integer), 부동 소수점(float point), 불(boolean)과 같은 단순한 자료형부터 메시지 안의 메시지를 품고 있는 간단한 데이터 구조, float32[ ] ranges나 Point32[10] points같은 메시지들이 나열된 배열 같은 구조, 헤더도 사용 가능
- 구성: 필드타입 필드네임 ex)fieldtype1 fieldname1

1) msg 파일: 토픽에서 사용되는 메시지 파일, *.msg, 필드타입 필드네임으로 구성
2) srv 파일: 서비스에서 사용되는 메시지 파일, *.srv
   3개의 하이픈(---)구분자로, 상위 메시지: 서비스 요청 메시지, 하위 메시지: 서비스 응답 메시지
3) action 파일: 액션에서 사용되는 메시지 파일, *.action
   3개의 하이픈(---)구분자 2군데 사용, feedback 메시지로 중간 결과값 전송

4.4 네임

- 그래프: 각 노드들의 연결 관계를 나타내고 화살표로 메시지(데이터)를 주고받는 관계 설명
- 이 때 필요한 노드, 토픽과 서비스에서 사용하는 메시지, 파라미터 모두 고유 네임을 가져야 함

```
int main(int argc, char **argv) //노드 메인 함수
{
        ros::init(argc, argv, "node1"); //노드명 초기화
        ros::NodeHandle nh; //노드 핸들 선언
        //퍼블리셔 선언, 토픽명 = bar
        ros::Publisher node1_pub = nh.advertise<std_msg::Int32>("bar",10);
}
```

Node	Relative(기본)	Global	Private
/node1	bar → /bar	/bar → /bar	~bar → /node1/bar
/wg/node2	bar → /wg/bar	/bar → /bar	~bar → /wg/node2/bar
/wg/node3	foo/bar → /wg/foo/bar	/foo/bar → /foo/bar	~foo/bar → /wg/node3/foo/bar

4.4 네임

- 같은 기능을 수행하는 관련 노드 두 번 실행해야 할 시, 노드의 이름을 변경하여 실행!

```
$rosrun camera_package camera_node //카메라 노드 실행
$rosrun rqt_image_view rqt_image_view //카메라 영상값을 image 토픽으로 전송시 rqt_image_view 통해 전송받음

//방법1) 리핑맵
$rosrun camera_package camera_nodeimage:=front/image //주고받는 토핑명 /front/image 변경
$rosrun rqt_image_view rqt_image_viewimage:=front/image

//방법2) 네임스페이스
$rosrun  camera_package camera_node__name:=front _device:=/dev/video0
$rosrun  camera_package camera_node__name:=left _device:=/dev/video1
$rosrun  camera_package camera_node__name:=right _device:=/dev/video2
$rosrun rqt_image_view rqt_image_view

$rosrun camera_package camera_node__ns:=back //정해진 노드 및 토픽 네임스페이스에 묶여 모든 네임 변경
$rosrun rqt_image_view rqt_image_view__ns:=back
```

4.5 좌표 변환(TF)

- 각 로봇의 관절(혹은 회전축 등을 갖는 바퀴) 및 물체의 위치를 상대 좌표 변환하여 기술

- 자세(pose) = 위치(position) + 방향(orientation)

  = x, y, z 벡터 + 쿼터니언 x, y, z, w (특정점을 v벡터를 중심으로 w만금 회전시킴)

  = transform.translation.x /transform.translation.y/transform.translation.z

  transform.rotation.x/transform.rotation.y/transform.rotation.z/transform.rotation.w

```
Header header            //변환된 시간을 기록
string child_frame_id    //하위 좌표 명시를 위한 메시지
Transform transform      //상대 위치와 방향 기술
```

4.6 클라이언트 라이브러리

- 다양한 언어로 작성 가능하게 해주는 소프트웨어 모듈

- 노드는 각각의 언어로 작성 가능, 노드 간의 메시지 통신을 통해 정보 교환

- <라이브러리 종류>

  더 빠르고 성능 중심: C++ → roscpp

  높은 생산성추구: Python → rospy

  인공지능 분야: 리스프(LISP) → roslisp

  안드로이드: 자바 → rosjava

  그 외: roscs, roseus, rosgo, roshask, rosnodejs, RobotOS.jl, roslua, PhaROS, rosR, rosruby, Unreal-Ros-Plugin 등

4.7 이기종 디바이스 간의 통신

- ROS는 메타 운영체제, 메시지 통신, 메시지, 네임, 좌표 변환, 클라이언트 라이브러리 등의 콘셉트를 통해 이기종 디바이스 간의 통신을 기본적으로 지원함

- ROS가 설치되어 사용하는 **운영체제의 종류와도 상관없고**, 사용하는 **프로그래밍 언어도 상관없이 ROS가 설치되어 각 노드가 개발**되었다면 각 **노드들 간의 통신 매우 쉽게** 이용 가능

- ROS를 설치할 수 없는 임베디드 시스템인 마이크로컨트롤러 기반의 디바이스이더라도 ROS의 메시지를 송수신만 받을 수 있도록 지원해준다면 로봇 제어에서 많이 사용되는 제어기 레벨에서도 메시지 통신이 가능하게 됨

4.8 파일 시스템

패키지 설치 2가지 방법
1) 바이너리 설치 $sudo apt-get install ros-kinetic-turtlebot3
2) 소스 코드 설치
$ cd ~/catkin_ws/src
$ git clone http://github.com/ROBOTIS-GIT/turtlebot3.git
$ cd ~/catkin_ws/
$ catkin_make

파일구성

- 패키지 = 1개 이상의 노드+다른 노드를 실행하기 위한 설정 파일들
- 메타패키지 = 공동 목적을 지닌 패키지들의 집합

- 각 패키지 구성:
  package.xml 파일: 패키지 정보를 담은 XML파일로 패키지 이름, 저작자, 라이선스, 의존성 패키지 등을 기술
  CMakeLists.txt 파일 : ROS 빌드시스템 캐킨이 이용하는 CMake 빌드 환경 기술
  이 외, 노드의 소스 코드 및 노드 간의 메시지 통신을 위한 메시지 파일 등

- 파일 시스템: 설치 폴더 + 사용자 작업 폴더
  설치폴더: 데스크톱 버전의 경우, /opt 폴더에 ros이름으로 폴더 생성 그 안에 roscore 포함 핵심 유틸리티와 rqt, RViz, 로봇 관련 라이브러리, 시뮬레이션, 내비게이션 등이 설치됨
  사용자 작업폴더: 사용자가 작성한 패키지와 공개된 다른 개발자의 패키지를 저장하고 빌드하는 공간 사용자가 원하는 곳에 폴더 생성 가능

4.8 파일 시스템

설치 폴더

- ROS 설치 폴더 경로: /opt/ros/[버전이름] 폴더
  ex)/opt/ros/noetic
- 파일 구성: /opt/ros/[버전이름] 폴더 아래
  bin, etc, include, lib, share 폴더와
  몇가지 환경설정 파일들
- 세부 내용: /bin : 실행가능한 바이너리 파일
  /etc : ROS 및 catkin 관련 설정 파일
  /include : 헤더 파일
  /lib : 라이브러리 파일
  /share : ROS 패키지
  env.* : 환경설정 파일
  setup.* : 환경설정 파일

작업 폴더

- 작업 폴더 경로: /home/사용자이름/catkin_ws/
  ex)/home/oroca/catkin_ws/
  (사용자 원하는 곳에 생성 가능 but
  리눅스 사용자 폴더 ~/catkin_ws/ 사용)
- 파일 구성: /home/사용자이름/catkin_ws/폴더
  아래 build, devel, src폴더로 구성
- 세부 내용: /build : 빌드 관련 파일
  /devel : msg, srv 헤더 파일과
  사용자 패키지 라이브러리, 실행 파일
  /src : 사용자 패키지

4.8 파일 시스템

사용자 패키지

- 사용자 소스코드 공간: ~/catkin_ws/src

- 세부 내용: /include : 헤더 파일
  /launch : roslaunch에 사용되는 launch파일
  /node : rospy용 스크립트
  /msg : 메시지 파일
  /src : 코드 소스 파일
  /srv : 서비스 파일
  CMakeLists.txt : 빌드 설정 파일
  package.xml : 패키지 설정 파일

4.9 빌드 시스템

- 빌드 시스템은 CMake 사용, 빌드 환경은 CMakeLists.txt 파일에 기술
- WHY CMake? 유닉스 계열인 리눅스, BSD, OS X, 윈도우 계열도 지원하여 ROS 패키지를 멀티 플랫폼에서 빌드할 수 있게 하기 위함
  +마이크로소프트 비주얼 스튜디오 +Qt 개발 쉽게 적용
  +캐킨빌드 시스템: ros와 관련된 빌드, 패키지 관리, 패키지 간 의존 관계 등 편리하게 사용가능
- 패키지 생성 명령어: `$catkin_create_pkg [패키지이름][의존하는 패키지1][의존하는패키지n]`

-The end-