

ROS 2

[02 – Development Environment]

한국폴리텍대학교 성남캠퍼스

1 WSL / Linux 설치

❖ Windows

- 컴퓨터에 대한 지식이 부족하더라도 쉽게 사용할 수 있도록 지원하여, 컴퓨터 사용자의 저변 확대에 기여
- 초기 Windows는 MS-DOS라는 운영체제 위에서 작동
 - ✓ MS-DOS는 Unix 계열로 시작
 - ✓ 발전 과정에서 실행파일(binary), 파일 시스템 관리 등의 별도의 체계 구축
 - ✓ 또 다른 Unix 계열인 Linux와 다른 형태로 발전하며 상호 호환성이 낮아졌음
- 과거, 네트워크가 발전하지 못한 시절은 윈도우가 강세, 이후 네트워크 고도화로 서버 환경이 중요해지면서 경제적 문제로 윈도우 서버로 서비스를 운영하기에 어려움이 있어 Linux 운영체제로 서비스 운영

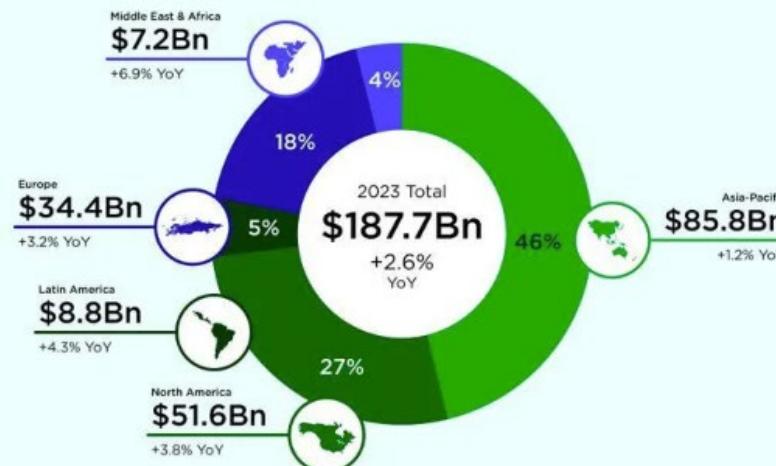
❖ Windows vs. Linux

- 대부분의 컴퓨터 사용자들은 업무를 수행하거나 게임을 플레이하기 위해 사용하고 있어 클라이언트 기반의 윈도우 시장이 큰 비중을 차지하고 있음
- 서비스 사용자와 서비스 개발자 환경은 차이가 발생할 수 밖에 없는 환경



The global games market in 2023

Per region with year-on-year growth rates



Source: @Newzoo | Global Games Market Report | July 2023
<https://newzoo.com/games-market-reports-forecasts>

WSL / Linux 설치

❖ What is WSL? (Windows Sub-system for Linux)

- 기존의 VM(Virtual Machine) 실행 시 발생하는 CPU, Memory 문제 및 VM을 위한 네트워크 설정에 대한 문제 없음
- Windows에서 GNU/Linux 계열의 표준 실행파일인 ELF(Executable and Linkable Format) binary 파일 실행 가능
- Windows 운영 체제에서 Linux 실행파일들을 함께 사용할 수 있도록 호환성을 추가하는 아키텍처임

- Microsoft WSL Docs

<https://learn.microsoft.com/ko-kr/windows/wsl/about>



WSL

❖ WSL 장점

- Windows 운영체제에서 경량화 가상화 기술을 사용하여 Linux 운영 체제를 구동할 수 있도록 지원
 - ✓ Windows 내부 시스템 기술로 Linux 구동 가능
 - ✓ 과거 가상화와 개념은 같으나 별도의 운영체제를 모두 실행하는 방식이 아닌 필요한 부분만 별도로 실행시키는 방식 사용
 - ✓ VM에 비해 기술적 제약이 거의 없고 빠르게 동작 가능

❖ WSL1 vs. WSL2

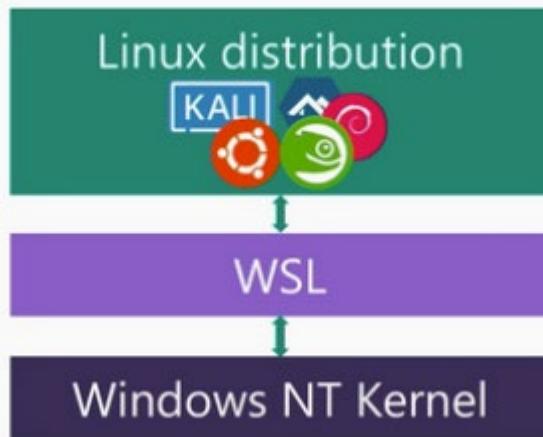
- 2020년 5월 Windows 10 업데이트에서 WSL2 발표
 - ✓ WSL1 : 고급 개발자 대상
 - ✓ WSL2 : 범용 개발에도 쓰일 수 있도록 발표

기능	WSL 1	WSL 2
Windows와 Linux 통합	✓	✓
빠른 부팅 시간	✓	✓
기존 Virtual Machines보다 작은 리소스 공간	✓	✓
현재 버전의 VMware 및 VirtualBox에서 실행	✓	✗
관리 VM	✗	✓
전체 Linux 커널	✗	✓
전체 시스템 호출 호환성	✗	✓
OS 파일 시스템 간 성능	✓	✗
체계적인 지원	✗	✓
IPv6 지원	✓	✓

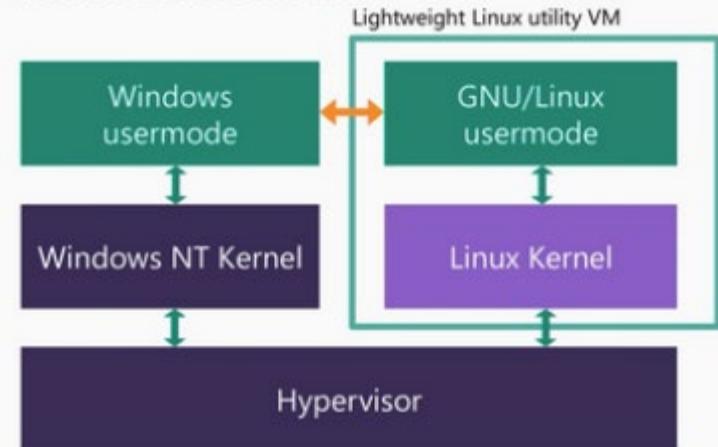
❖ WSL1 vs. WSL2

- WSL1은 Windows에서 Linux 시스템을 WSL이 Emulating하여 처리하는 방식
- 모든 Linux 명령어를 처리하는 데 한계가 존재하고 처리속도가 느림
- WSL2는 실제 Linux Kernel이 탑재되어 있어 Windows 커널과 병렬로 처리하여 모든 Linux 명령어를 처리할 수 있음
- Docker 지원 (<https://www.docker.com/>)
- Docker 개요

WSL architecture



WSL 2 architecture overview

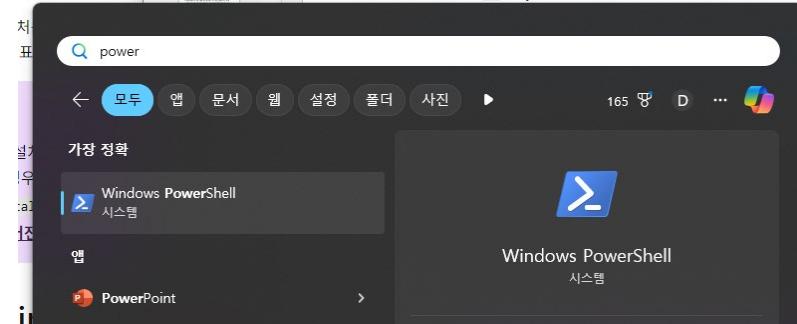


WSL / Linux 설치

❖ WSL 설치

- <https://learn.microsoft.com/ko-kr/windows/wsl/install>

- PowerShell을 관리자 모드로 실행



```
PowerShell
wsl --install
```

A screenshot of a PowerShell window. The title bar says 'PowerShell'. In the main pane, the command 'wsl --install' is being typed. A '복사' (Copy) button is visible in the top right corner of the input field.

① 참고

위의 명령은 WSL이 전혀 설치되지 않은 경우에만 작동합니다. WSL 도움말 텍스트를 실행하고 `wsl --install` 보는 경우 실행 `wsl --list --online` 하여 사용 가능한 배포판 목록을 확인하고 실행 `wsl --install -d <DistroName>` 하여 배포판 설치를 시도하세요. WSL을 제거하려면 [WSL의 레거시 버전 제거](#) 또는 [Linux 배포판 등록 취소 또는 제거](#)를 참조하세요.

❖ 실습환경

1. 운영체계

- Ubuntu 22.04.4 LTS(Jammy Jellyfish)
- <https://learn.microsoft.com/ko-kr/windows/wsl/install>

2. ROS

- ROS 2 Humble
- <https://docs.ros.org/en/humble/index.html>

3. IDE

- Visual Studio Code

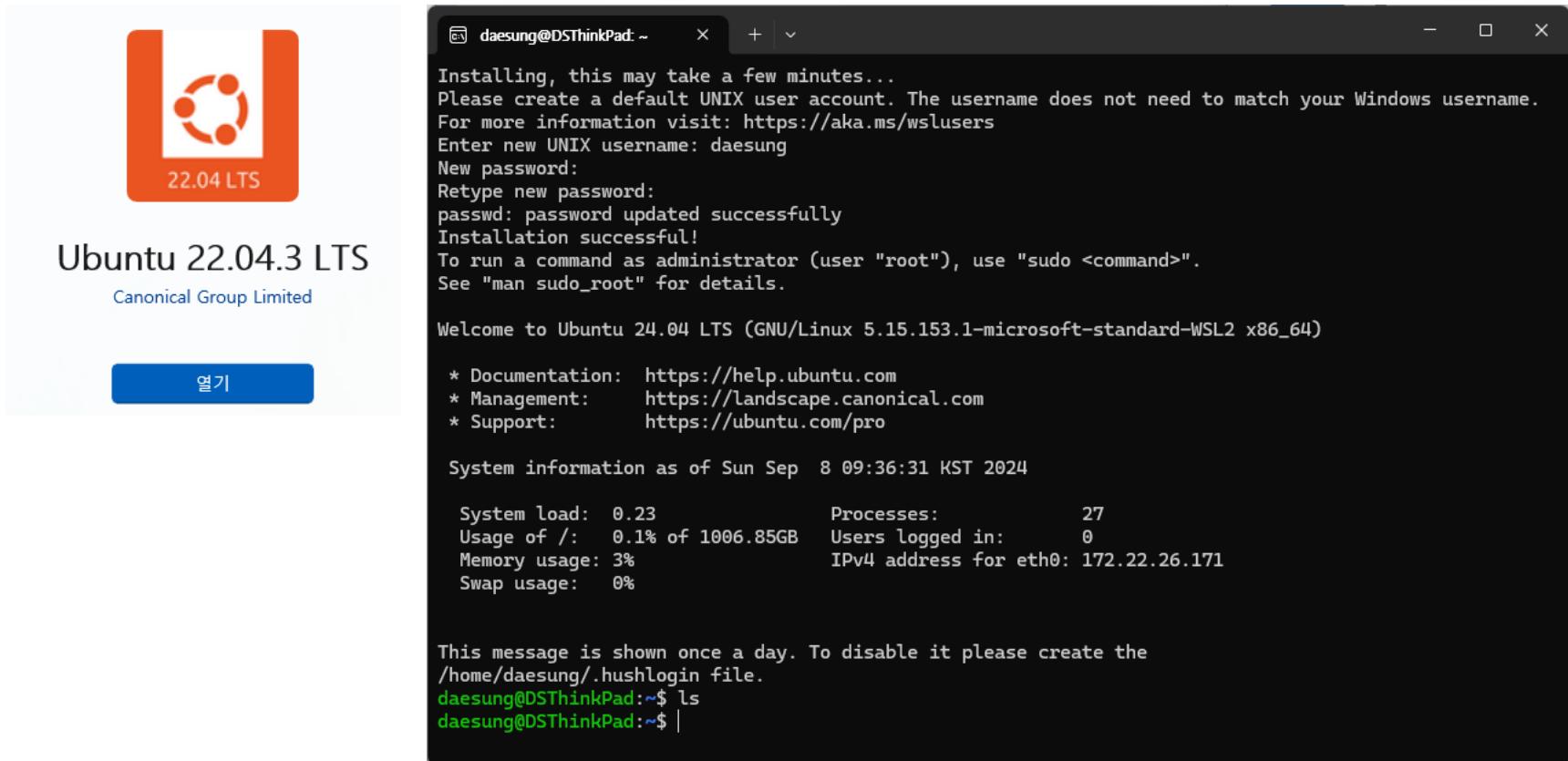
4. Programming Language

- Python, C++

WSL / Linux 설치

❖ Ubuntu 22.04 LTS 설치

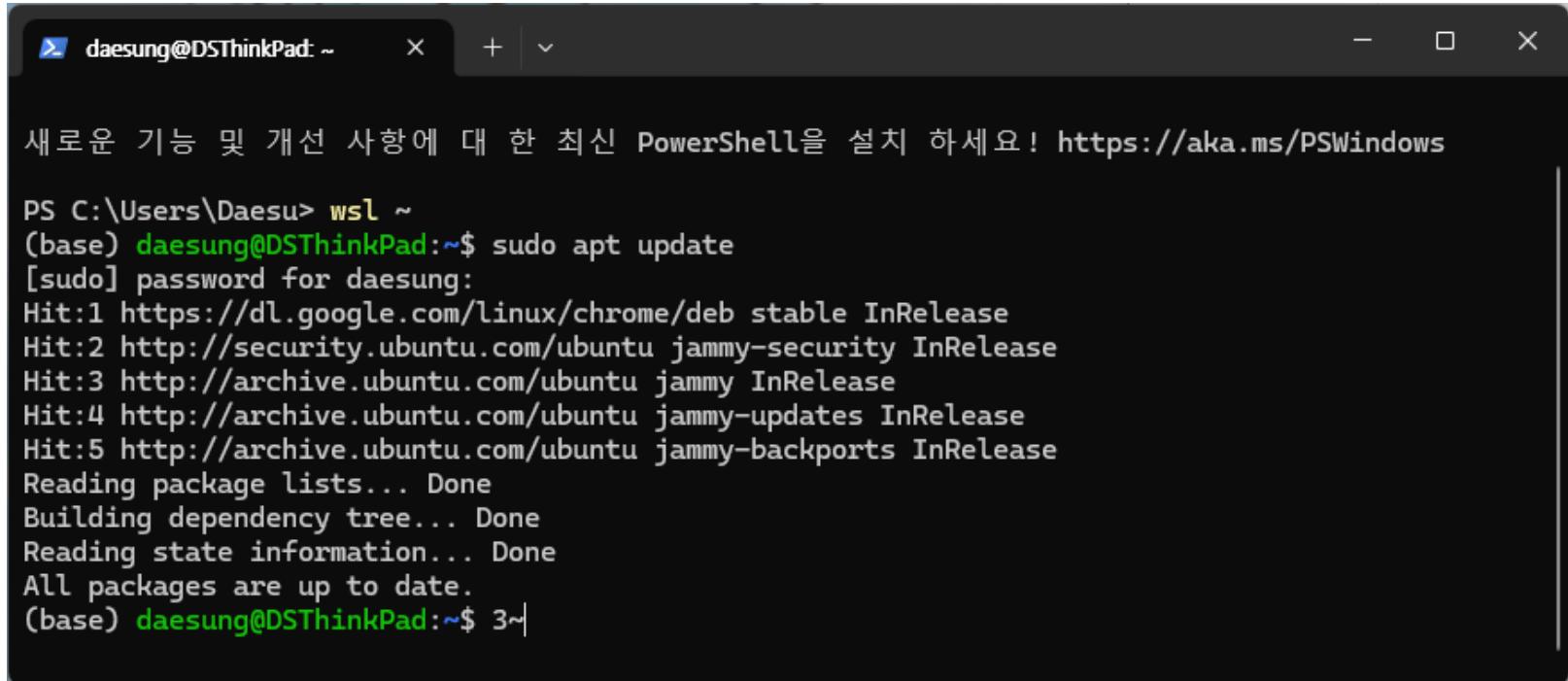
- Microsoft Store에서 검색 후 설치



WSL / Linux 설치

❖ Ubuntu 22.04 LTS 실행 및 업데이트

- Windows PowerShell 실행
 - ✓ wsl ~ 입력으로 Linux 실행



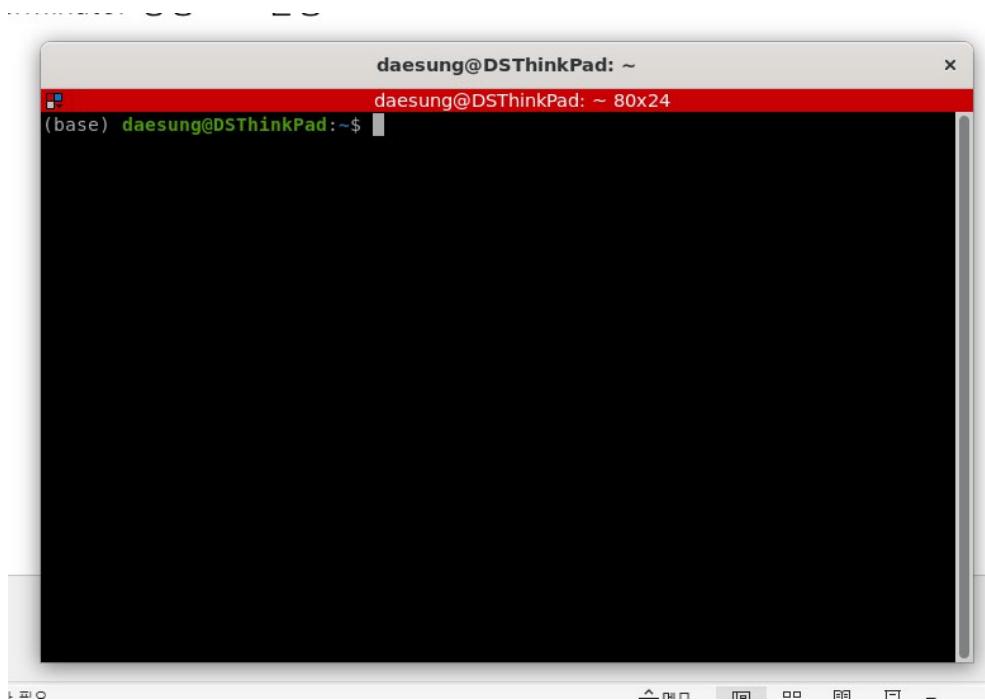
The screenshot shows a Windows terminal window with a dark theme. The title bar reads "daesung@DSThinkPad: ~". The window contains the following text:

```
새로운 기능 및 개선 사항에 대 한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows

PS C:\Users\Daesu> wsl ~
(base) daesung@DSThinkPad:~$ sudo apt update
[sudo] password for daesung:
Hit:1 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
(base) daesung@DSThinkPad:~$ 3~|
```

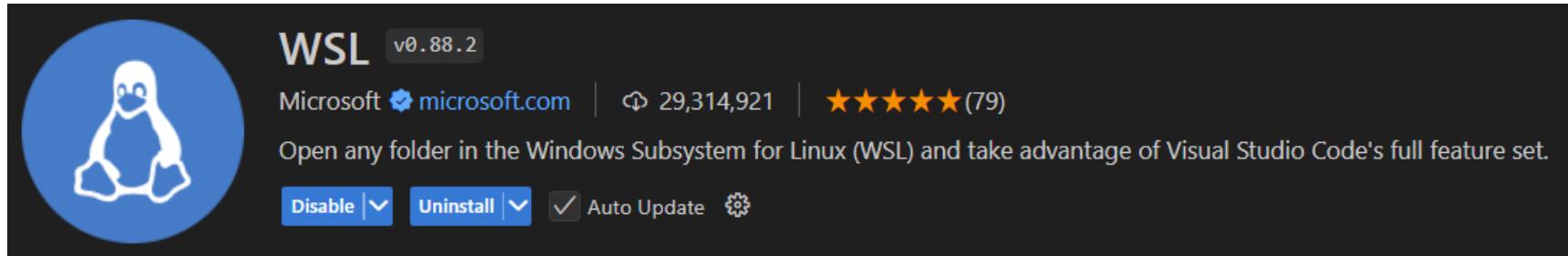
❖ Terminator 설치

- Linux Terminal에 다음과 같은 명령 입력
 - ✓ sudo apt install terminator
- terminator 명령으로 실행

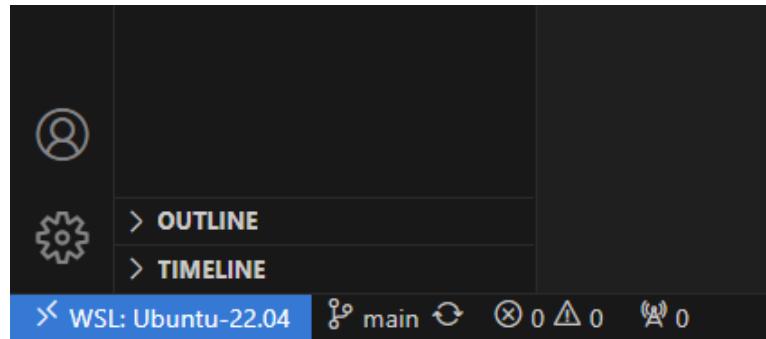


WSL / Linux 설치

❖ VS Code Extension 설치



- 설치 후 좌측 하단에 WSL 선택 후 터미널 실행



2 Linux 기초

❖ What is Linux?

- 시스템 동작을 위한 운영체제(OS, Operating System) 중 한 가지
- AT&T 벨 연구소에서 1960년대에 개발
- CLI (Command Line Interface) 기반
- 현재 GUI (Graphic User Interface)도 지원

❖ Linux is

- 유닉스는 리눅스의 원형이며 리눅스는 유닉스에서 파생된 OS
 - ✓ Mac OS, Windows 역시 유닉스에서 파생된 OS임
- Linux는 무료로 사용할 수 있는 Open Source이며 서버 분야에서 가장 많이 사용되고 있음
- 다중사용자 및 다중처리를 지원하며, 많은 참여자의 기여로 인해 다양한 환경에 대한 이식성이 좋음
 - ✓ PowerPC, ARM 등
- 다양한 리눅스 배포판이 있음
 - ✓ Redhat, Fedora, Ubuntu, SUSE

❖ Linux 명령어

- date
 - ✓ 현재 시간과 날짜를 알려주는 명령어
- ls
 - ✓ 현재 디렉토리의 리스트를 조회하는 명령어
 - ✓ List를 의미함
 - ✓ 옵션
 - -a : 숨김 파일을 포함한 전체 파일 조회
 - -l : 리스트 형태로 상세내용 출력
 - -h : 사용자가 볼 수 있는 파일 크기 단위로 보기
 - -t : 시간 순서대로 정렬
 - 상기 옵션의 조합을 통해 파일 리스트 확인 가능
(예 : -al → 디렉토리 내 모든 파일 및 폴더 리스트 확인)

❖ Linux 명령어

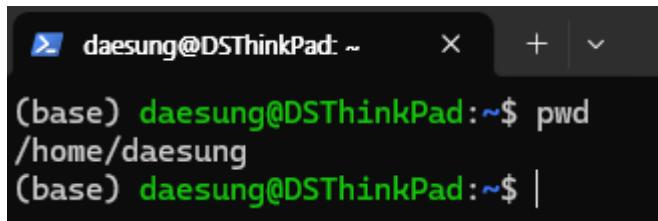
- cd
 - ✓ Change directory라는 뜻을 가지고 있음
 - ✓ cd [이동할 디렉토리 경로]
 - ✓ 계층구조 이해
 - 리눅스 시스템은 윈도우의 시스템과 차이점이 있음
 - 윈도우는 C 드라이브, D드라이브 형태로 되어 있고, 하나의 물리적/논리적 디스크로 나뉘어져 있음
 - 리눅스는 특정 폴더에도 디스크를 할당할 수 있음
 - ✓ cd .. 상위 디렉토리로 이동
 - ✓ cd ~ root 디렉토리로 이동

```
(base) daesung@DSThinkPad:/$ ls
bin  dev  home  lib    lib64  lost+found  mnt  proc  run  snap  sys  usr
boot  etc  init  lib32  libx32  media      opt  root  sbin  srv  tmp  var

```

❖ Linux 명령어

- pwd
 - ✓ 현재 작업 중인 디렉토리를 절대경로로 출력



A screenshot of a terminal window titled 'daesung@DSThinkPad: ~'. The window shows the command 'pwd' being run and its output '/home/daesung'. The terminal has a dark background with white text.

```
daesung@DSThinkPad: ~
(base) daesung@DSThinkPad:~$ pwd
/home/daesung
(base) daesung@DSThinkPad:~$ |
```

- cp
 - ✓ 특정 파일을 복사하는 명령어
 - ✓ cp [arg] [복사할 파일] [복사될 파일]
 - ✓ 옵션
 - -r : 폴더까지 복사해주는 명령어
 - -f : 같은 파일이 있으면 강제로 덮어씌움
 - -v : verbose의 약자로 상세한 복사내역을 표시

❖ Linux 명령어

- mv
 - ✓ 특정 파일을 이동하거나 이름을 바꾸는 명령어
 - ✓ cp 와 옵션이 거의 똑같지만 이동할 때 디렉토리까지 모두 이동하기 때문에 –r 옵션은 사용하지 않음
- rm
 - ✓ 특정 파일이나 디렉토리를 지우는 명령어이므로 주의 필요
 - ✓ 옵션
 - -r : 디렉토리를 포함해서 삭제
 - -v : 삭제한 내용에 대해 표시
 - -f : 강제로 삭제
- clear
 - ✓ 화면 내용 삭제

❖ Linux 명령어

- mv
 - ✓ 특정 파일을 이동하거나 이름을 바꾸는 명령어
 - ✓ cp 와 옵션이 거의 똑같지만 이동할 때 디렉토리까지 모두 이동하기 때문에 –r 옵션은 사용하지 않음
- rm
 - ✓ 특정 파일이나 디렉토리를 지우는 명령어이므로 주의 필요
 - ✓ 옵션
 - -r : 디렉토리를 포함해서 삭제
 - -v : 삭제한 내용에 대해 표시
 - -f : 강제로 삭제
- clear
 - ✓ 화면 내용 삭제

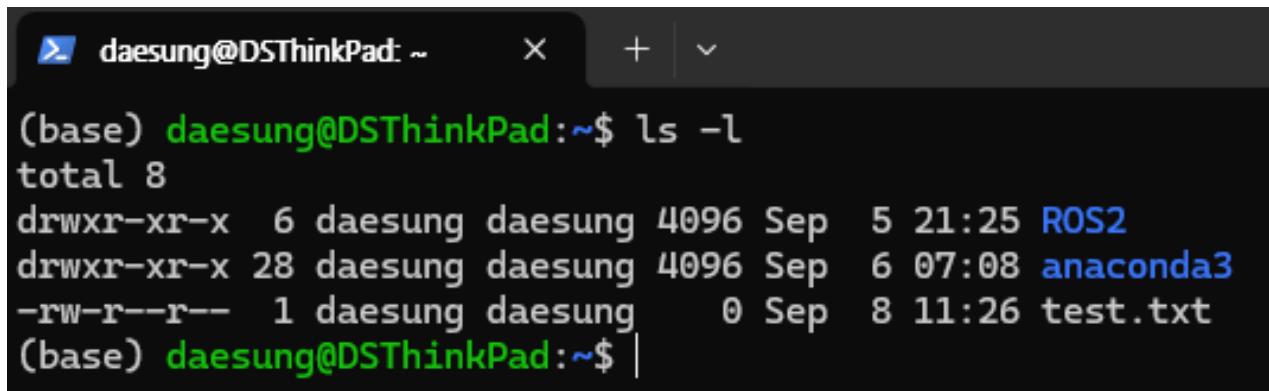
❖ Linux 명령어

- ps
 - ✓ 실행 중인 프로세스 상태와 정보를 출력
 - ✓ ps 명령어만 입력하게 되면 현재 사용자의 프로세스만 나타남
 - ✓ 대부분 ps -ef 옵션을 통해 실행 중인 모든 프로세스의 자세한 정보 취득
- kill
 - ✓ 프로세스를 종료 시킬 때 사용하는 명령어
 - ✓ 명령어 실행 후 정상적으로 종료되었는지 ps를 통해 확인 필요
- passwd
 - ✓ root 계정에서 passwd [사용자명]을 입력하면 해당 사용자의 비밀번호를 변경할 수 있음
- cat / more
 - ✓ cat [파일명] 파일의 내용 표시
 - ✓ More [파일명] 파일의 내용을 표시할 수 있을 만큼 분할하여 표시

❖ Linux 명령어

- chmod

- ✓ 리눅스는 많은 사용자 계정을 생성하여 사용하므로 permission 설정이 필요
- ✓ Permission은 3가지 영역으로 구분되며 각각의 권한 표시
- ✓ ABC
 - A: 사용자 / B: 그룹 / C: 사용자와 그룹 이외
 - 각 영역은 3-bit의 데이터로 표시
 - 상위 bit부터 Read, Write, Execute Permission으로 값을 할당
- ✓ chmod 744 test.txt의 경우 사용자(rwx), 그룹(r), 사용자와 그룹 이외(r)로 설정



```
(base) daesung@DSThinkPad:~$ ls -l
total 8
drwxr-xr-x  6 daesung daesung 4096 Sep  5 21:25 ROS2
drwxr-xr-x 28 daesung daesung 4096 Sep  6 07:08 anaconda3
-rw-r--r--  1 daesung daesung     0 Sep  8 11:26 test.txt
(base) daesung@DSThinkPad:~$ |
```

❖ Linux 명령어

- sudo
 - ✓ 관리자 권한으로 실행할 경우 sudo 사용
 - ✓ 처음 실행할 경우 비밀번호 입력 필요

```
(base) daesung@DSThinkPad:~$ sudo apt update
[sudo] password for daesung:
Hit:1 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1988 kB]
Fetched 2245 kB in 3s (706 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
(base) daesung@DSThinkPad:~$ |
```

❖ Linux 명령어

- apt

- ✓ Ubuntu/Raspbian 처럼 Debian 계열의 리눅스에서는 apt를 사용해서 패키지 관리
- ✓ apt update : 서버에서 새로운 패키지 업데이트
- ✓ apt install : 서버에서 새로운 패키지 설치
- ✓ apt upgrade : 서버에서 새로운 업데이트 설치

```
daesung@DSThinkPad:~$ apt upgrade
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
(base) daesung@DSThinkPad:~$ sudo apt up
update  upgrade
(base) daesung@DSThinkPad:~$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
(base) daesung@DSThinkPad:~$ |
```

❖ Linux 명령어

- wget
 - ✓ 파일 다운로드 받기
 - ✓ wget [option] [URL]
 - ✓ 참고 <https://myminju.tistory.com/68>
- man
 - ✓ 명령어에 대한 설명(manual)
 - ✓ man [명령어]
- find
 - ✓ 파일 찾기
 - ✓ find [경로] [옵션]

❖ Linux 명령어

- gzip
 - ✓ gzip 명령어는 파일을 압축하는 명령어
 - ✓ gzip [option] [File]
 - ✓ gzip -h
- tar
 - ✓ tar 명령어는 폴더까지 압축하는 명령어
 - ✓ tar –cvzf filename.tar.gz 파일경로/ 사용

3 실습환경 구축

❖ Anaconda 설치-1

1. update

- sudo apt update

2. curl 패키지 설치

- sudo apt install curl -y

3. Anaconda 설치 ([버전 참고](#))

- curl --output anaconda.sh https://repo.anaconda.com/archive/Anaconda3-2022.10-Linux-x86_64.sh
- sha256sum anaconda.sh
- bash anaconda.sh

실습환경 구축

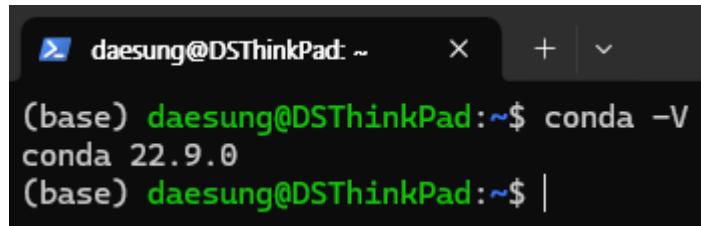
❖ Anaconda 설치-2

4. conda 명령어 환경변수 추가

- vs code를 이용하여 .bahsrc 파일 수정
 - sudo code ~/.bashrc
- 마지막 줄에 다음과 같은 라인 추가
 - export PATH=~/anaconda3/bin:~/anaconda3/condabin:\$PATH
- .bashrc 다시 실행하여 업데이트 내용 적용
 - Source ~/.bashrc

5. 설치 확인

- conda -V



```
daesung@DSThinkPad: ~
(base) daesung@DSThinkPad:~$ conda -V
conda 22.9.0
(base) daesung@DSThinkPad:~$ |
```

❖ ROS2의 20가지 특징

(1) Platforms

- ROS2부터는 3대 운영체제인 Linux, Windows, macOS를 모두 지원
- ROS2 Jazzy Jalisco 기준으로 보았을 때 Linux는 Ubuntu Noble (24.04), Windows는 Windows 10 버전, macOS는 Mojave (10.14)버전을 지원하고 있음
- Linux, macOS는 ROS1 부터 지원하고 있었는데 이번 ROS2에서는 Microsoft가 ROS 2 TSC로 들어오고 Windows용 패키지 및 테스트, Visual Studio Code Extension for ROS까지 준비하는 등 굉장한 노력을 기울여 Windows 사용자들도 ROS2를 쉽게 사용할 수 있게 되었음

<https://microsoft.github.io/Win-RoS-Landing-Page/>

❖ ROS2의 20가지 특징

(2) Real-time

- ROS 2는 Real-time을 지원
- 단, 선별된 하드웨어 사용, 리얼타임 운영체제 사용, DDS의 RTPS(Real-time Publish-Subscribe Protocol)와 같은 통신 프로토콜을 사용, 매우 잘 짜여진 리얼타임 코드 사용을 전제로 함

<https://www.apex.ai/roscon2019>

❖ ROS2의 20가지 특징

(3) Security

- ROS1에서는 항상 보안이 문제였음. 노드를 관리하는 ROS master의 하나의 IP와 포트만 노출되면 모든 시스템을 죽일 수 있는 관계로, 보안 입장에서는 TCPROS 매우 취약
- 이러한 취약점은 상용 로봇에 ROS를 도입할 수 없게 만드는 첫번째 이유이자 가장 큰 걸림돌이 됨
- ROS2에서는 TCP 기반의 통신은 OMG(Object Management Group)에서 산업용으로 사용 중인 DDS(Data Distribution Service)를 도입하여 DDS-Security이라는 DDS 보안 기능을 ROS에 적용하여 보안에 대한 이슈를 통신단부터 해결
- 또한 ROS 커뮤니티에서는 SROS 2(Secure Robot Operating System 2)라는 툴을 개발하였고 보안 관련 RCL 서포트 및 보안관련 프로그래밍에 익숙지 않은 로보틱스 개발자를 위해 보안을 위한 툴킷을 만들어 배포

❖ ROS2의 20가지 특징

(4) Communication

- ROS2는 리얼타임 퍼블리시와 서브스크라이브 프로토콜인 RTPS(Real Time Publish Subscribe)를 지원하는 통신 미들웨어 DDS를 사용
- DDS는 OMG(Object Management Group)에 의해 표준화가 진행되고 있으며, 상업적인 용도에도 적합하다는 평가가 지배적
- DDS에서는 IDL(Interface Description Language)를 사용하여 메시지 정의 및 직렬화를 더 쉽게, 더 포괄적으로 다룰 수 있음
- 통신 프로토콜로는 RTPS을 채용하여 실시간 데이터 전송을 보장하고 임베디드 시스템에도 사용할 수 있음
- DDS에 대한 내용은 별도로 다룰 예정임

❖ ROS2의 20가지 특징

(5) Middleware interface

- DDS는 다양한 기업에서 통신 미들웨어 형태로 제공
 - ✓ ROS2를 지원하는 업체는 ADLink, Eclipse Foundation, Eprosima, Gurum Network, RTI로 총 5 곳
 - ✓ DDS 제품명으로는 ADLINK의 OpenSplice, Eclipse Foundation의 Cyclone DDS, Eprosima의 Fast DDS, Gurum Network의 Gurum DDS, RTI의 Connext DDS가 있음
- ROS2에서는 이러한 미들웨어를 유저가 원하는 사용 목적에 맞게 선택하여 사용할 수 있도록 ROS Middleware(RMW)형태로 지원
- RMW는 여러 DDS 구현을 지원하기 위하여 API의 추상화 인터페이스로 지원하고 있다.

❖ ROS2의 20가지 특징

(6) Node manager (discovery)

- ROS1에서의 필수 실행 프로그램으로는 roscore가 있으며, 이를 실행시키면 ROS Master, ROS Parameter Server, rosout logging node가 실행
- ROS Master는 ROS 시스템의 각각 독립되어 실행되는 노드들의 정보를 관리하여 서로 연결해야 하는 노드들에게 상대방 노드의 정보를 건네 주어 연결할 수 있게 해주는 매우 중요한 중계 역할 수행
- 했었다. 이 때문에 ROS 1에서는 노드 사이의 연결을 위해 네임 서비스를 마스터에서 실행해야 했고, 이 ROS Master가 연결이 끊어지거나 멈추면 모든 시스템이 마비되는 단점이 있었음
- ROS2에서는 roscore가 없어지고 3가지 프로그램이 각각 독립 수행
- 특히, ROS Master의 경우 DDS를 사용함에 따라 노드를 DDS의 Participant 개념으로 취급하게 되어 완전히 삭제

❖ ROS2의 20가지 특징

(6) Node manager (discovery)

- ROS1에서의 필수 실행 프로그램으로는 roscore가 있으며, 이를 실행시키면 ROS Master, ROS Parameter Server, rosout logging node가 실행
- ROS Master는 ROS 시스템의 각각 독립되어 실행되는 노드들의 정보를 관리하여 서로 연결해야 하는 노드들에게 상대방 노드의 정보를 건네 주어 연결할 수 있게 해주는 매우 중요한 중계 역할 수행
- 했었다. 이 때문에 ROS 1에서는 노드 사이의 연결을 위해 네임 서비스를 마스터에서 실행해야 했고, 이 ROS Master가 연결이 끊어지거나 멈추면 모든 시스템이 마비되는 단점이 있었음
- ROS2에서는 roscore가 없어지고 3가지 프로그램이 각각 독립 수행
- 특히, ROS Master의 경우 DDS를 사용함에 따라 노드를 DDS의 Participant 개념으로 취급하게 되어 완전히 삭제

❖ ROS2의 20가지 특징

(7) Languages

- ROS 1: C++03, Python 2.7
- ROS 2: C++17, Python 3.8

❖ ROS2의 20가지 특징

(8) Build system

- ROS 2에서는 새로운 빌드 시스템인 ament을 사용
- ament는 ROS1에서 사용되는 빌드 시스템인 catkin의 업그레이드 버전
- ROS1의 catkin이 CMake만을 지원했던 반면, ament는 CMake를 사용하지 않는 Python 패키지 관리도 가능
 - ✓ ROS 1: rosbuild → catkin (CMake)
 - ✓ ROS 2: ament (CMake), Python setuptools (Full support)

❖ ROS2의 20가지 특징

(9) Build tools

- ROS1의 경우 여러 가지 다른 도구, 즉 catkin_make, catkin_make_isolated 및 catkin_tools가 지원
- ROS2에서는 알파, 베타, 그리고 Ardent 릴리스까지 빌드 도구로 ament_tools가 이용되었고 지금에 와서는 colcon을 추천하고 있음
- colcon은 ROS2 패키지를 작성, 테스트, 빌드 등 ROS2 기반의 프로그램을 할 때 빼놓을 수 없는 툴로 작업흐름을 향상시키는 CLI 타입의 명령어 도구
- 추후 이어지는 상세히 다룰 예정

❖ ROS2의 20가지 특징

(10) Build options

- ROS 2에서는 빌드 관련 내용들이 모두 변경되면서 빌드 옵션에도 새로운 변화가 생겼으며, 대표적인 점 3가지는 다음과 같으며, 향후 상세히 다룰 예정
 - ✓ Multiple workspace
 - ✓ No non-isolated build
 - ✓ No devel space

❖ ROS2의 20가지 특징

(11) Version control system

- ROS는 수 많은 소스 코드 공여자로부터 만들어가는 코드의 집합이기 때문에 개인은 물론 소속도 정말 다양하고 각 코드들의 리포지토리도 제각각
- 예를 들어 어느 패키지는 GitHub를 이용하고 어떤 것은 Bitbucket를 이용
- 사용하는 버전 관리 시스템(Version Control System, VCS)도 Git, Mercurial, Subversion, Bazaar 등 다양
 - ✓ ROS 1: rosdep → wstool, rosinstall (*.rosinstall)
 - ✓ ROS 2: vcstool (*.repos)

❖ ROS2의 20가지 특징

(12) Client library

- ROS 기반 프로그래밍은 ROS 구조에서 유저 코드 영역(user land)을 다루는 것을 의미함
- ROS 클라이언트 라이브러리(ROS Client Library)가 있고, 이 클라이언트 라이브러리는 앞서 설명한 미들웨어(middleware interface)를 사용
- 유저는 개발 목적에 따라 C/C++, Python, Java, Node.js 등을 사용
- ROS 2에서는 ROS 클라이언트 라이브러리를 RCL(ROS Client Library)이라는 이름으로 제공
- 프로그래밍 언어별로 rclcpp, rclc, rclpy, rcljava, rclobjc, rclada, rclgo, rclnodejs 등으로 제공

❖ ROS2의 20가지 특징

(13) Life cycle

- 로봇 개발에 있어서 로봇의 현재 상태를 파악하고 현재 상태에서 다른 상태로 변경되는 상태전이 제어는 수십년간 로봇공학에서도 주요 연구 주제로 다루었던 중요한 부분
- 태스크 수행 측면에서 현재의 상태 파악과 천이는 멀티 태스크 수행에서 빠질 수 없는 중요한 부분일 것이고 복수의 로봇 복수의 복합 태스크, 서비스 수행과 같은 상위 레벨의 프로그램 일수록 더 중요하게 다루어지는 부분
- ROS2에서는 패키지의 각 노드들의 현재 상태를 모니터링하고 상태를 제어 가능한 lifecycle을 클라이언트 라이브러리에 포함

❖ ROS2의 20가지 특징

(14) Multiple nodes

- ROS1의 초기에는 하나의 프로세스에서 여러 노드를 실행 할 수 없었음
- nodelet라는 기능 추가로 ROS1에 하드웨어 리소스가 제한적이거나 노드 간에 수 많은 메시지를 보내야 할 때 유용하게 사용
- ROS 2에서 nodelet이 사용되지는 않고 RCL에 포함되어 있음
- 컴포넌트(components)라고 부르며 ROS 2에서는 이 컴포넌트를 사용하여 동일한 실행 파일에서 복수의 노드를 수행
- 노드의 실행 파일 수준은 더 세분화 시킬 수 있으며 프로세스 내 통신 IPC(intra-process communication)기능을 이용하여 ROS 2의 통신 오버 헤드를 제거. 더 효율적인 ROS 2 응용 프로그램을 작성 가능

❖ ROS2의 20가지 특징

(15) Threading model

- ROS1에서 개발자는 단일 스레드 실행 또는 다중 스레드 실행 중 하나만 선택
- ROS 2에서는 더 세분화 된 실행 모델(executor)을 C++과 Python에서 사용할 수 있으며 사용자가 정의한 실행기도 제공되는 RCL API를 이용하여 쉽게 구현할 수 있음

❖ ROS2의 20가지 특징

(16) Messages (topic, service, action)

- ROS2에서도 기존 ROS1의 메시지(Messages)과 마찬가지로 단일 데이터 구조를 메시지라고 정의하며 정해진 또는 사용자가 정의한 메시지를 사용할 수 있으며, 각 패키지 이름과 마찬가지로 이름과 각 지정된 형식으로 메시지를 고유하게 식별할 수 있음
- 기존과 마찬가지로 Topic, Service, Action 등에서 사용하며 기존과 비슷한 형태로 사용 가능
- ROS2에서는 OMG(Object Management Group)에서 정의된 IDL(Interface Description Language)을 사용하여 메시지 정의 및 직렬화를 더 쉽게, 더 포괄적으로 다룰 수 있게 되었음

❖ ROS2의 20가지 특징

(17) Command Line Interface

- 대부분의 CLI 타입의 명령어 사용법은 기존 ROS1과 매우 비슷
- ROS2의 CLI 형태의 명령어는 ROS2 TSC 멤버이자 Ubuntu 개발 업체인 Canonical이 담당하고 있어서 신뢰도가 높음
 - ✓ ROS 1: rostopic list
 - ✓ ROS 2: ros2 topic list

❖ ROS2의 20가지 특징

(18) roslaunch

- ROS의 실행 시스템은 대표적으로 `run`과 `launch`가 있음
 - ✓ `run`은 단일 프로그램 실행,
 - ✓ `launch`는 사용자 지정 프로그램 실행을 수행
- ROS1과 ROS2의 차이점은 다양한 파일 사용
- ROS1에서는 `roslaunch` 파일이 특정 `XML` 형식을 사용
- ROS2에서는 `XML` 형식 이외에도 `Python`이 새롭게 채용되어 조건문 및 Python 모듈을 추가로 사용하여 보다 복잡한 논리와 기능 사용 가능

❖ ROS2의 20가지 특징

(19) Graph API

- ROS1에서 `rqt_graph` 툴을 자주 사용
- ROS에서 각 노드와 토픽, 메시지 등이 고유의 이름을 가지고 있고 매팅이 이루어져 각 노드와 노드간의 토픽, 메시지의 관계를 그래프화
- ROS 1에서 불편한 점은 시작할 때만 노드와 토픽 간의 매팅이 이루어져서 실시간으로 변화되는 그래프를 볼 수 없었음
- ROS2에서는 노드 시작할 때 뿐만 아니라 실행 도중에 다시 매팅도 가능하며, 그 결과를 바로 그래프로 표현할 수 있음

❖ ROS2의 20가지 특징

(20) Embedded Systems

- 로봇 개발에 있어서 실시간성을 담보 받으며 모터 및 센싱을 제어하는 부분은 매우 중요
- 실시간성이라는 것은 상위 소프트웨어에서 다루기에는 제약이 많으며, Embedded Systems 안에서 해결 하는 것이 적합
- ROS2에서는 기존 시리얼 통신, 블루투스 및 와이파이 통신을 지원하거나 RTOS (Real-Time Operating System)를 사용하고 기존 DDS 대신 eXtremely Resource Constrained Environments (DDS-XRCE)를 사용 하는 등 임베디드 보드에서 직접 ROS 프로그래밍을 하여 하드웨어 펌웨어로 구현된 노드를 실행할 수도 있음

❖ ROS2 Humble 설치

Ubuntu ROS2 Humble install

Ubuntu (deb packages)

Table of Contents

- Resources
- Set locale
- Setup Sources
- Install ROS 2 packages
- Environment setup
 - Sourcing the setup script
- Try some examples
 - Talker-listener
- Next steps after installing
- Using the ROS 1 bridge
- Additional RMW implementations (optional)
- Troubleshooting
- Uninstall

설치 안내에 따라 Humble 설치 후 테스트

❖ ROS2 Humble 데모 테스트

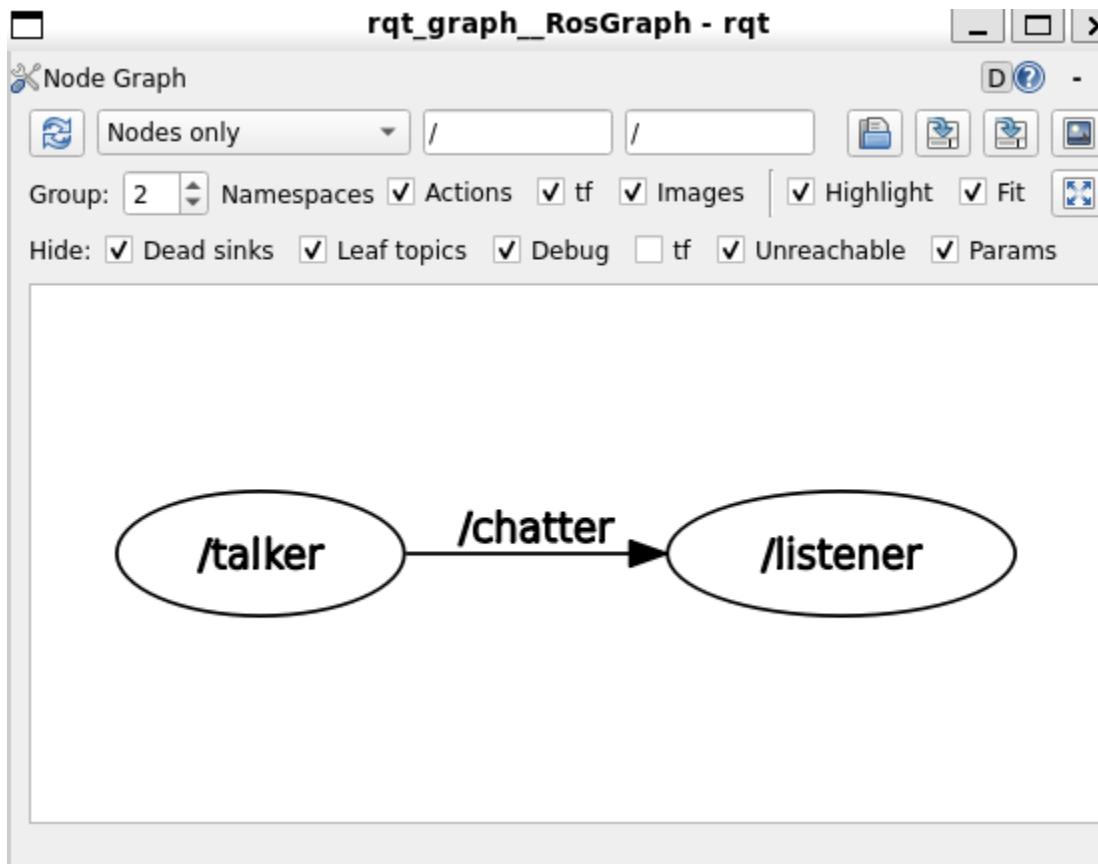
The screenshot shows two terminal windows side-by-side. Both windows have a title bar 'daesung@DSThinkPad: ~' and a size indicator '80x14' or '80x13'. The top window displays the output of a publisher node, showing multiple [INFO] messages from '[talker]' at various timestamps between 1725776823 and 1725776835. Each message contains the string 'Hello World: <some number>'. The bottom window displays the output of a subscriber node, showing multiple [INFO] messages from '[listener]' at the same timestamps, each containing the string 'I heard: [Hello World: <the same number>]'.

```
[INFO] [1725776823.738708568] [talker]: Publishing: 'Hello World: 69'  
[INFO] [1725776824.738712062] [talker]: Publishing: 'Hello World: 70'  
[INFO] [1725776825.738509695] [talker]: Publishing: 'Hello World: 71'  
[INFO] [1725776826.738657189] [talker]: Publishing: 'Hello World: 72'  
[INFO] [1725776827.738579972] [talker]: Publishing: 'Hello World: 73'  
[INFO] [1725776828.738492836] [talker]: Publishing: 'Hello World: 74'  
[INFO] [1725776829.738682904] [talker]: Publishing: 'Hello World: 75'  
[INFO] [1725776830.738783479] [talker]: Publishing: 'Hello World: 76'  
[INFO] [1725776831.739027441] [talker]: Publishing: 'Hello World: 77'  
[INFO] [1725776832.738513645] [talker]: Publishing: 'Hello World: 78'  
[INFO] [1725776833.738650125] [talker]: Publishing: 'Hello World: 79'  
[INFO] [1725776834.738554410] [talker]: Publishing: 'Hello World: 80'  
[INFO] [1725776835.738568720] [talker]: Publishing: 'Hello World: 81'  
[]  
[INFO] [1725776824.739887223] [listener]: I heard: [Hello World: 70]  
[INFO] [1725776825.739759984] [listener]: I heard: [Hello World: 71]  
[INFO] [1725776826.740005292] [listener]: I heard: [Hello World: 72]  
[INFO] [1725776827.739649449] [listener]: I heard: [Hello World: 73]  
[INFO] [1725776828.739331350] [listener]: I heard: [Hello World: 74]  
[INFO] [1725776829.739683646] [listener]: I heard: [Hello World: 75]  
[INFO] [1725776830.740615424] [listener]: I heard: [Hello World: 76]  
[INFO] [1725776831.740101375] [listener]: I heard: [Hello World: 77]  
[INFO] [1725776832.739476176] [listener]: I heard: [Hello World: 78]  
[INFO] [1725776833.740192402] [listener]: I heard: [Hello World: 79]  
[INFO] [1725776834.739222150] [listener]: I heard: [Hello World: 80]  
[INFO] [1725776835.740136796] [listener]: I heard: [Hello World: 81]
```

실습환경 구축

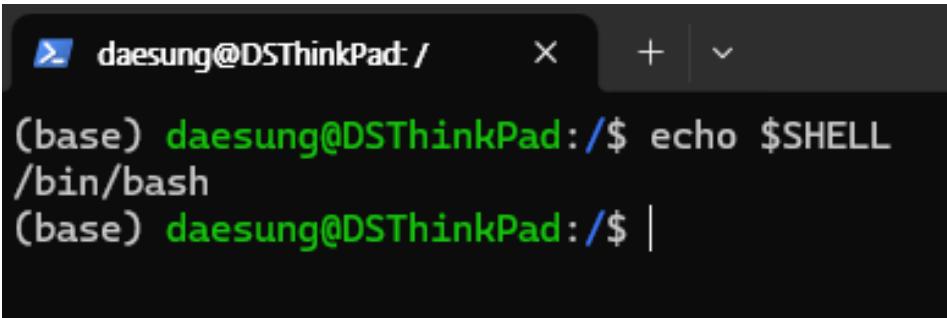
❖ ROS2 Humble 데모 테스트

- rqt_graph



❖ bashrc

- Shell
 - ✓ Shell은 운영체제의 일부
 - ✓ PC가 실행된 이후 메모리에 상주하는 핵심 커널과 사용자 사이를 연결해 주는 프로그램
 - ✓ 사용자가 직접 커널에 명령어를 입력하는 CLI (Command Line Interface) 방식
 - ✓ 유닉스 초기기부터 사용자가 직접 명령을 입력하고 명령을 OS에 전달하는 역할 수행
 - ✓ Ubuntu에서 사용하는 shell은 배수(bash)

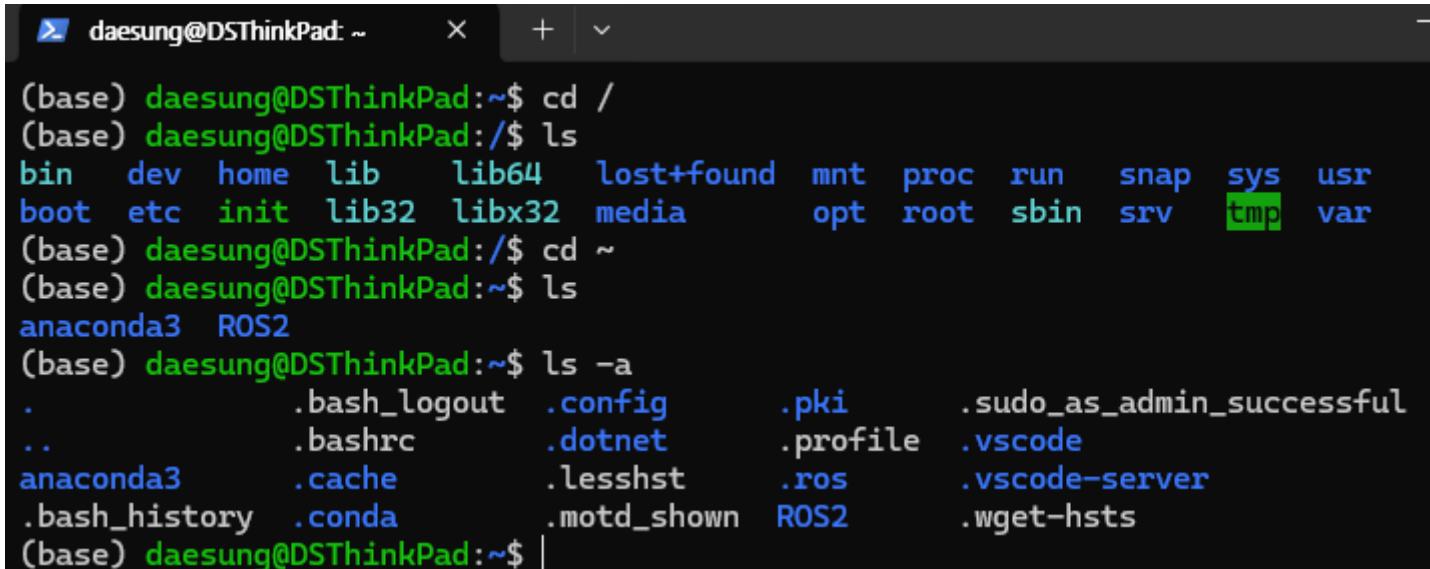


```
daesung@DSThinkPad: /$ echo $SHELL
/bin/bash
daesung@DSThinkPad:/$ |
```

실습환경 구축

❖ ~/.bashrc

- Ubuntu의 Shell인 bash의 설정을 저장하는 파일 중 로그인한 사용자 별로 지정한 설정을 저장해 두는 파일이 .bashrc이며, 로그인 후 자동으로 실행
- 파일의 위치는 /home/사용자명 에 위치하고 있으며 dot(.) 으로 시작하는 파일은 숨김 파일로 ls –a 명령으로 확인할 수 있음



```
(base) daesung@DSThinkPad:~$ cd /
(base) daesung@DSThinkPad:/$ ls
bin  dev  home  lib    lib64  lost+found  mnt  proc  run  snap  sys  usr
boot etc  init  lib32  libx32  media      opt  root  sbin  srv   tmp  var
(base) daesung@DSThinkPad:/$ cd ~
(base) daesung@DSThinkPad:~$ ls
anaconda3  ROS2
(base) daesung@DSThinkPad:~$ ls -a
.              .bash_logout  .config        .pki          .sudo_as_admin_successful
..             .bashrc       .dotnet        .profile      .vscode
anaconda3     .cache        .lessht        .ros          .vscode-server
.bash_history  .conda       .motd_shown  ROS2         .wget-hsts
(base) daesung@DSThinkPad:~$ |
```

실습환경 구축

❖ ~/.bashrc

- ROS2 Humble을 실행하기 위해서는 다음과 같은 명령을 실행해야 동작함
 - ✓ Source /opt/ros/humble/setup.bash
- 로그인 시 자동으로 ROS를 실행하도록 vs code를 이용해 .bashrc에 저장

```
(base) daesung@DSThinkPad:/opt/ros/humble$ cd ~
(base) daesung@DSThinkPad:~$ ls
anaconda3  ROS2
(base) daesung@DSThinkPad:~$ code .bashrc
(base) daesung@DSThinkPad:~$ |
```

- .bashrc에 다음과 같이 추가

```
134
135 echo "ROS2 Humble is activated!"
136 source /opt/ros/humble/setup.bash|
```

실습환경 구축

❖ ~/.bashrc

- .bashrc를 다시 실행하면 다음과 같은 출력과 ROS2 Humble 실행

```
(base) daesung@DSThinkPad:/opt/ros/humble$ cd ~  
(base) daesung@DSThinkPad:~$ ls  
anaconda3  ROS2  
(base) daesung@DSThinkPad:~$ code .bashrc  
(base) daesung@DSThinkPad:~$ source .bashrc  
ROS2 Humble is activated!  
(base) daesung@DSThinkPad:~$
```

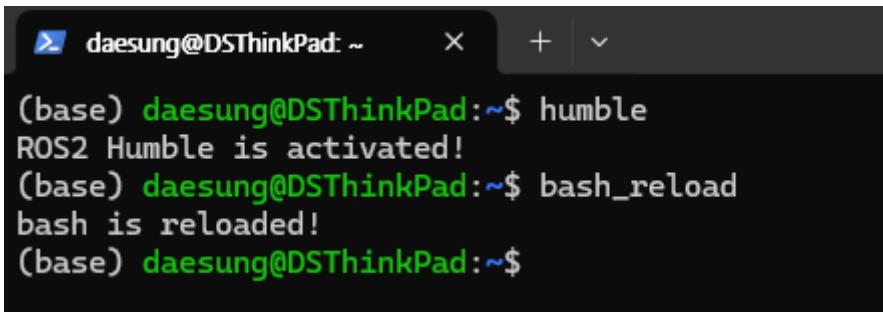
VS Code를 이용해 추가

```
134  
135 echo "ROS2 Humble is activated!"  
136 source /opt/ros/humble/setup.bash|
```

❖ alias 설정

- alias 설정을 통해 명령어 등록 및 실행
 - ✓ alias command_name="values"
- Humble 실행 및 .bashrc 재 실행도 alias로 지정
 - ✓ 공백 주의

```
134  
135 alias humble="source /opt/ros/humble/setup.bash; echo \"ROS2 Humble is activated!\""  
136 alias bash_reload="source ~/.bashrc; echo \"bash is reloaded!\""  
137
```



A screenshot of a terminal window titled 'daesung@DSThinkPad: ~'. The window shows two commands being run: 'humble' and 'bash_reload'. Both commands output a message indicating success: 'ROS2 Humble is activated!' and 'bash is reloaded!' respectively.

```
(base) daesung@DSThinkPad:~$ humble  
ROS2 Humble is activated!  
(base) daesung@DSThinkPad:~$ bash_reload  
bash is reloaded!  
(base) daesung@DSThinkPad:~$
```

❖ ROS2 Domain 설정

- ROS2는 DDS(Data Distribution Service)를 이용

데이터 분산 서비스(Data Distribution Service, DDS)는 실시간 시스템의 실시간성(real-time), 규모가변성(scalable), 안전성 (dependable), 고성능 (high performance)를 가능하게 하는 Object Management Group(OMG) 표준 출판/구독 (Publish/Subscribe) 네트워크 커뮤니케이션 미들웨어이다

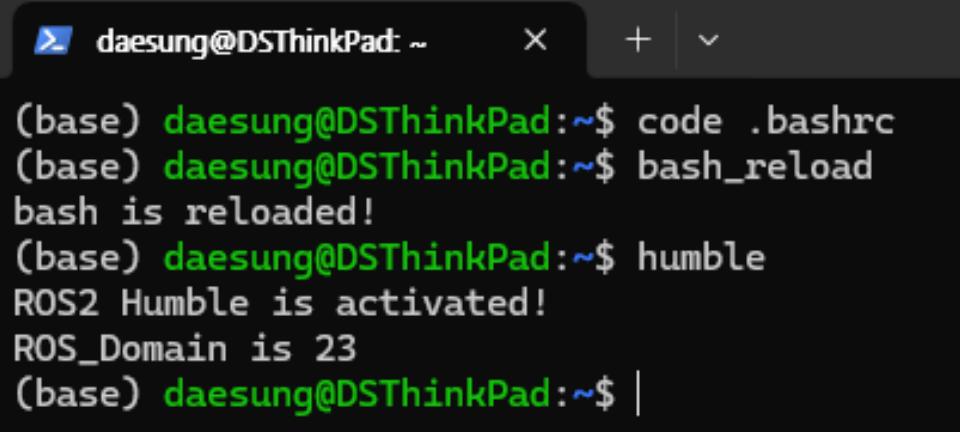
[위키백과 DDS]

실습환경 구축

❖ ROS2 Domain 설정

- DDS 적용을 통해 메시지의 발행/구독 시스템을 사용
- 하나의 AP(Access Point)에서 여러 사람이 동시에 ROS2를 사용하는 경우 같은 이름의 토픽 사용으로 충돌이 발생할 수 있음. 이러한 환경에서 채널 분리를 위해 Domain ID를 설정하여 사용할 수 있음

```
alias humble="source /opt/ros/humble/setup.bash; export ROS_DOMAIN_ID=23;
echo \"ROS2 Humble is activated!\";
echo \"ROS_Domain is 23\""
```



The screenshot shows a terminal window titled 'daesung@DSThinkPad: ~'. The user runs the command 'code .bashrc' to edit the bash configuration file. After saving and reloading the bash session with 'bash_reload', the user activates the ROS2 Humble distribution by running 'humble'. The terminal displays the message 'ROS2 Humble is activated!' and 'ROS_Domain is 23', confirming the successful setup.

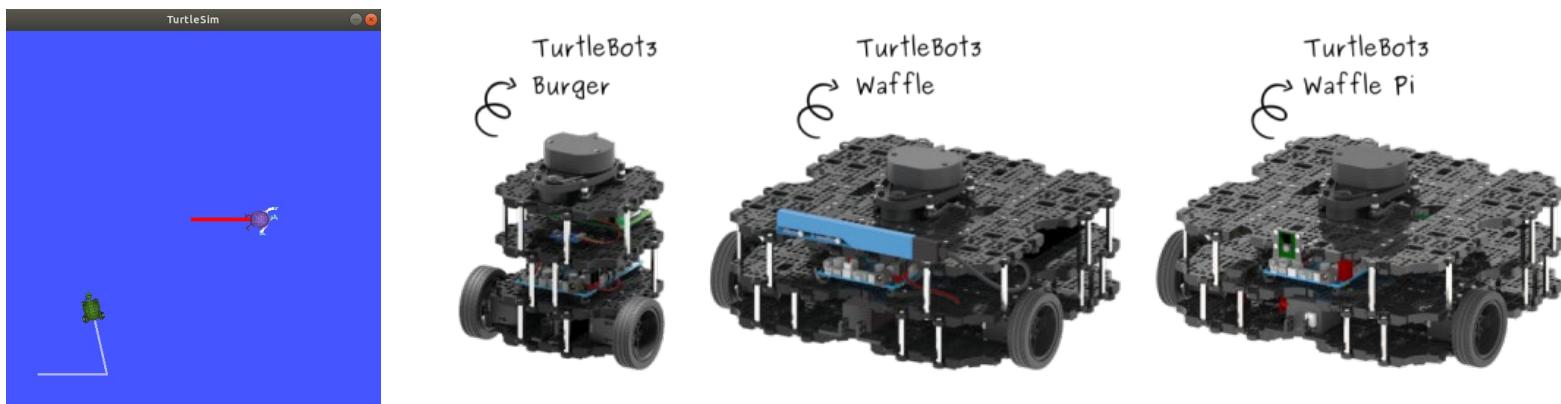
```
(base) daesung@DSThinkPad:~$ code .bashrc
(base) daesung@DSThinkPad:~$ bash_reload
bash is reloaded!
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ |
```

4 ROS2 Humble 시작하기

■ ROS2 Humble Basic

❖ Turtlesim

- 간단한 시뮬레이션 및 테스트를 위한 패키지
 - ✓ ROS에서 실행 가능한 최소한의 단위를 노드(node)라 칭함
 - ✓ 다수의 노드와 여러 설정을 모아 둔 것을 패키지라 칭함
- 서비스, 토픽, 액션, 디버그에 필요한 기능을 Turtlesim을 활용하여 테스트



ROS2 Humble 시작하기

■ Turtlesim 실행

❖ ROS2 실행

- 앞서 설정해 놓은 alias command를 이용해 ROS 실행

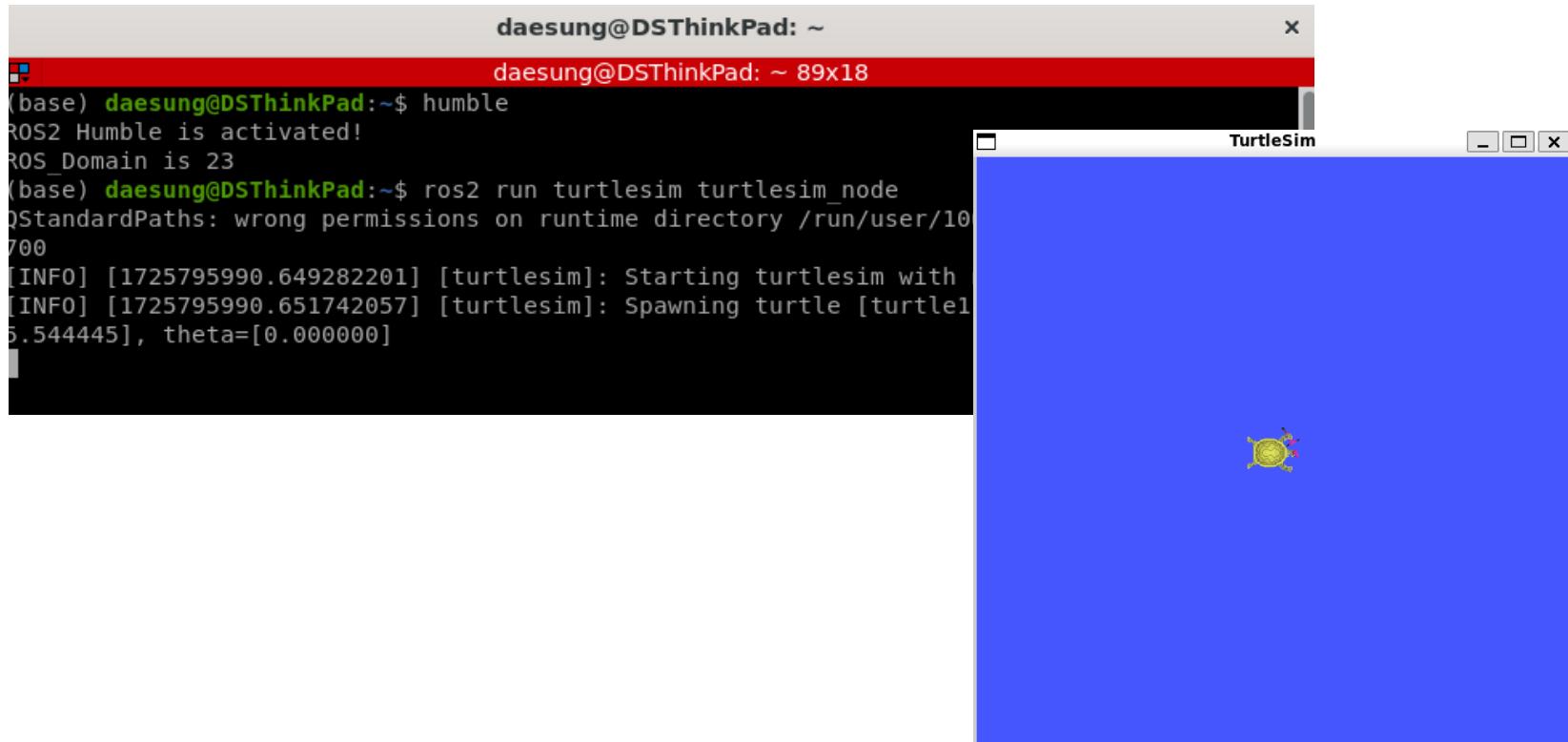
```
daesung@DSThinkPad: ~
daesung@DSThinkPad: ~ 89x18
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$
```

ROS2 Humble 시작하기

■ Turtlesim 실행

❖ Turtlesim_node 실행

- ros2 run turtlesim turtlesim_node

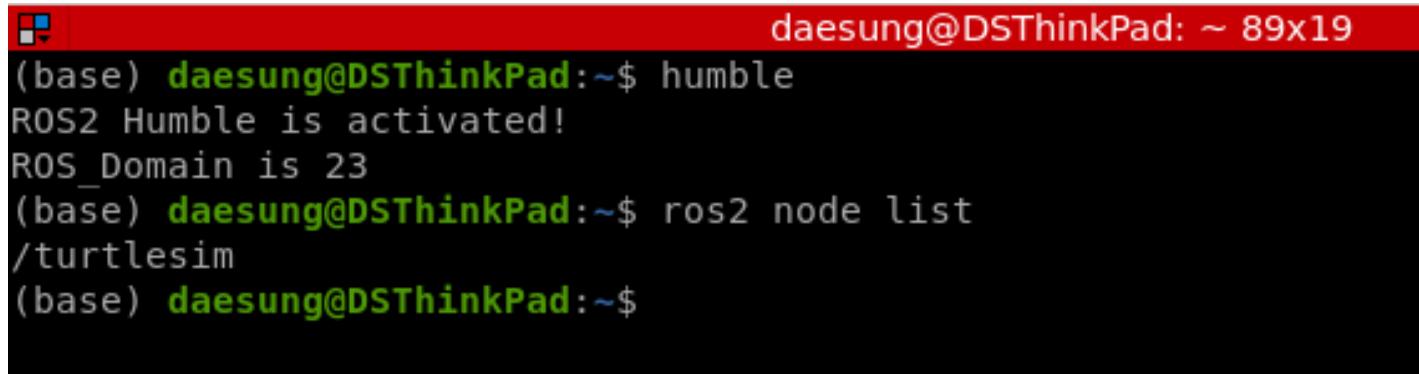


ROS2 Humble 시작하기

■ Turtlesim 실행

❖ 실행중인 노드 목록

- ros2 node list



```
daesung@DSThinkPad: ~ 89x19
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ ros2 node list
/turtlesim
(base) daesung@DSThinkPad:~$
```

■ Turtlesim 실행

❖ 실행중인 노드 정보

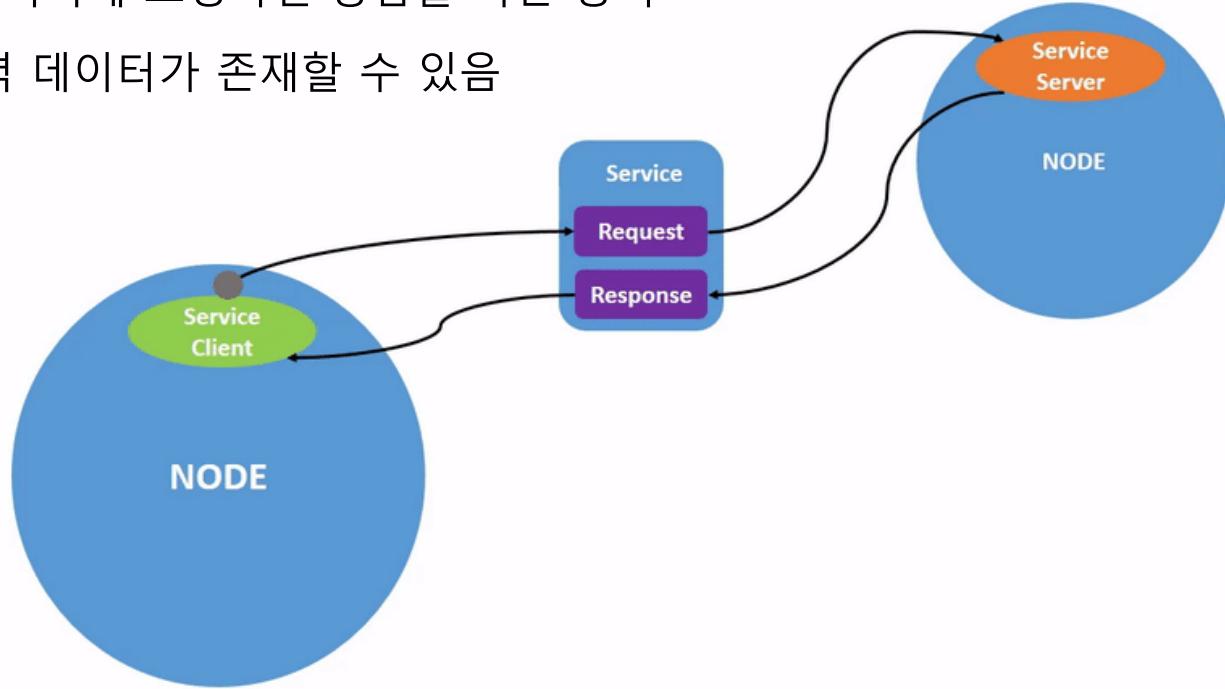
- ros2 node info /turtlesim

```
daesung@DSThinkPad: ~ 89x40
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ ros2 node list
/turtlesim
(base) daesung@DSThinkPad:~$ ros2 node info /turtlesim
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:
  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:
```

■ Turtlesim 실행

❖ ROS Service

- 두개의 노드가 데이터를 주고 받는 방식
- 클라이언트가 서버에 요청하면 응답을 하는 방식
- 입력 또는 출력 데이터가 존재할 수 있음



ROS2 Humble 시작하기

■ Turtlesim 실행

❖ ROS Service

- ros2 service list

```
daesung@DSThinkPad: ~ 89x40
(base) daesung@DSThinkPad:~$ ros2 service list
/clear
/kill
/reset
/spawn
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
(base) daesung@DSThinkPad:~$
```

■ Turtlesim 실행

❖ ROS Service

- ros2 service type /turtle1/teleport_absolute

```
daesung@DSThinkPad: ~ 89x40
(base) daesung@DSThinkPad:~$ ros2 service list
/clear
/kill
/reset
/spawn
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
(base) daesung@DSThinkPad:~$ ros2 service type /turtle1/teleport_absolute
turtlesim/srv/TeleportAbsolute
(base) daesung@DSThinkPad:~$
```

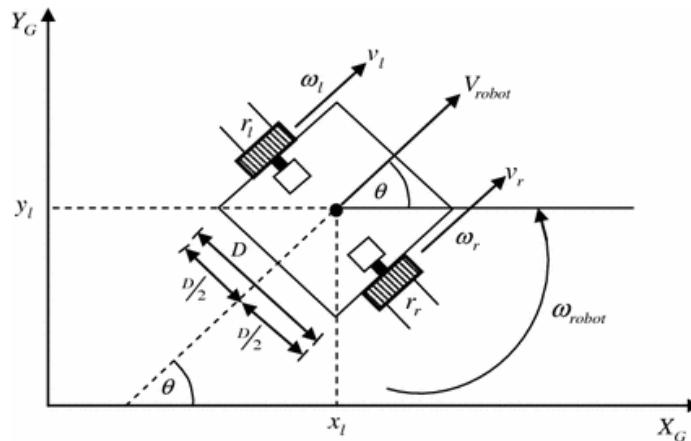
ROS2 Humble 시작하기

Turtlesim 실행

❖ ROS Service data

- ros2 interface show turtlesim/srv/TeleportAbsolute

```
(base) daesung@DSThinkPad:~$ ros2 interface show turtlesim/srv/TeleportAbsolute
float32 x
float32 y
float32 theta
---
(base) daesung@DSThinkPad:~$
```



Degrees → radians

$$\times \text{ by } \frac{\pi}{180}$$

Radians → degrees

$$\times \text{ by } \frac{180}{\pi}$$

ROS2 Humble 시작하기

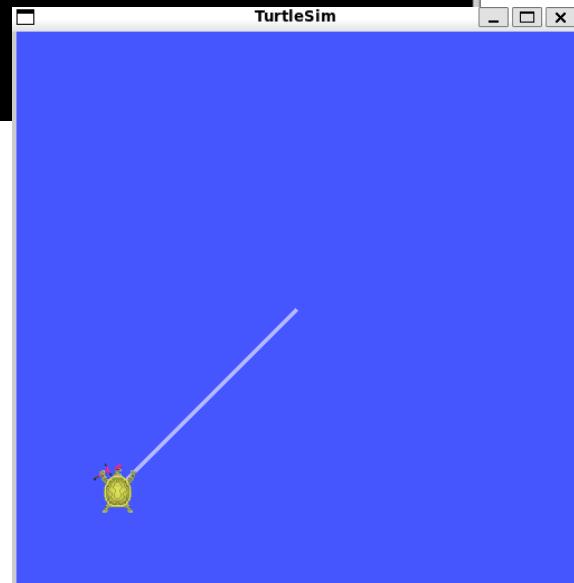
■ Turtlesim 실행

❖ ROS Service call

- ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 2, y: 2, theta: 1.57}"

```
(base) daesung@DSThinkPad:~$ ros2 service call /turtle1/teleport_absolute turtlesim/srv/TeleportAbsolute "{x: 2, y: 2, theta: 1.57}"
requester: making request: turtlesim.srv.TeleportAbsolute_Request(x=2.0, y=2.0, theta=1.57)

response:
turtlesim.srv.TeleportAbsolute_Response()
```



- **ros2 service call [Service name] [Service definition] "data"**

ROS2 Humble 시작하기

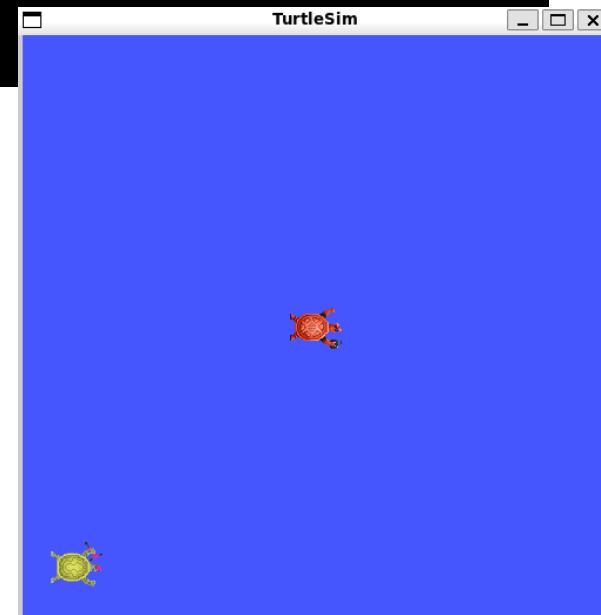
■ Turtlesim 실행

❖ ROS Service call

- ros2 service call /spawn turtlesim/srv/Spawn "{x: 1, y: 1, theta: 0, name: ""}"

```
(base) daesung@DSThinkPad:~$ ros2 service call /spawn turtlesim/srv/Spawn "{x: 1, y: 1, theta: 0, name: ""}"
requester: making request: turtlesim.srv.Spawn_Request(x=1.0, y=1.0, theta=0.0, name='None')

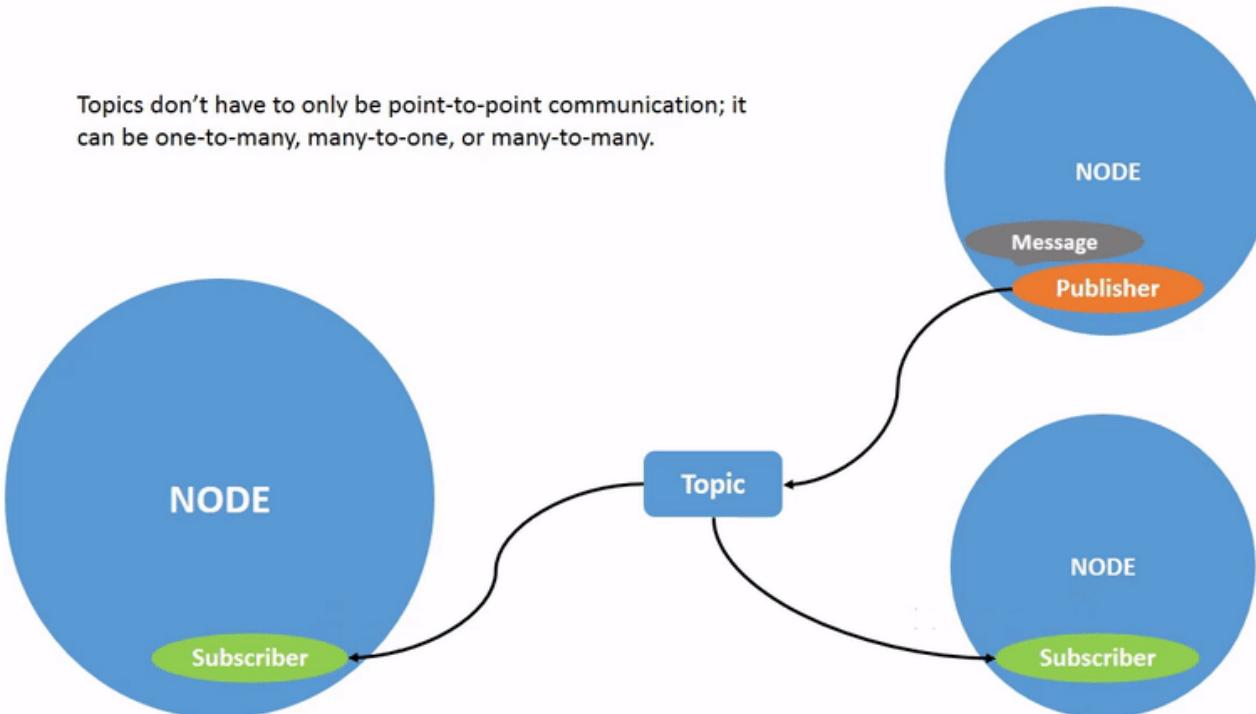
response:
turtlesim.srv.Spawn_Response(name='None')
```



■ Turtlesim 실행

❖ ROS Topic

- Topic은 Publish와 Subscribe으로 구성



ROS2 Humble 시작하기

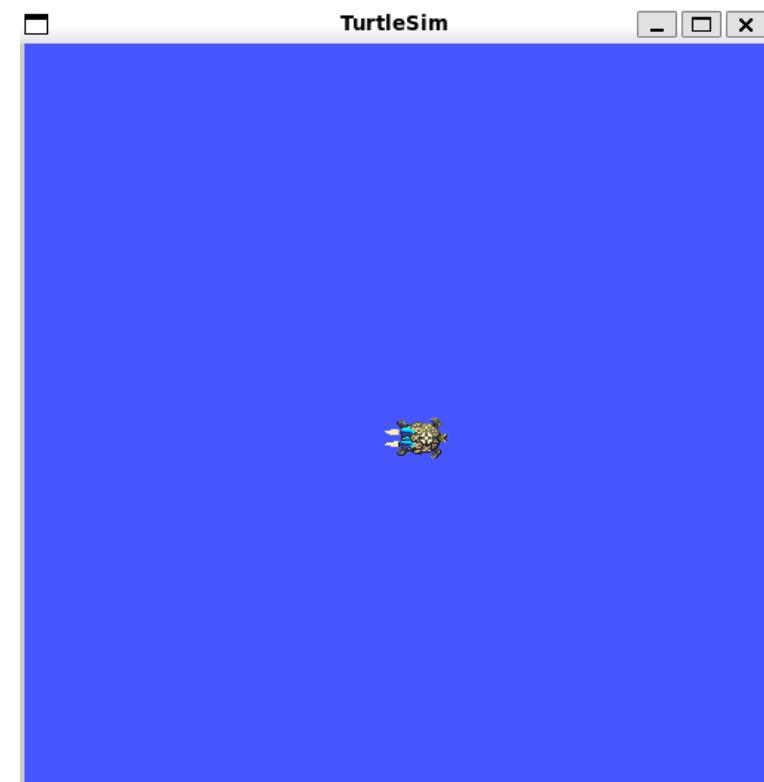
■ Turtlesim 실행

❖ ROS Topic

- Turtlesim 실행 후 topic list 조회

```
daesung@DSThinkPad: ~
daesung@DSThinkPad: ~ 61x13
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ ros2 run turtlesim turtlesim_node
QStandardPaths: wrong permissions on runtime directory /run/user/1000/, 0755 instead of 0700
[INFO] [1725799900.510379729] [turtlesim]: Starting turtlesim
with node name /turtlesim
[INFO] [1725799900.513816222] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
[

daesung@DSThinkPad: ~ 61x12
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
(base) daesung@DSThinkPad:~$
```



■ Turtlesim 실행

❖ ROS Topic

- Turtlesim 실행 후 topic list 조회 및 type 확인

```
daesung@DSThinkPad: ~ 61x12
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
(base) daesung@DSThinkPad:~$ ros2 topic type /turtle1/pose
turtlesim/msg/Pose
(base) daesung@DSThinkPad:~$
```

ROS2 Humble 시작하기

■ Turtlesim 실행

❖ ROS2 Topic info

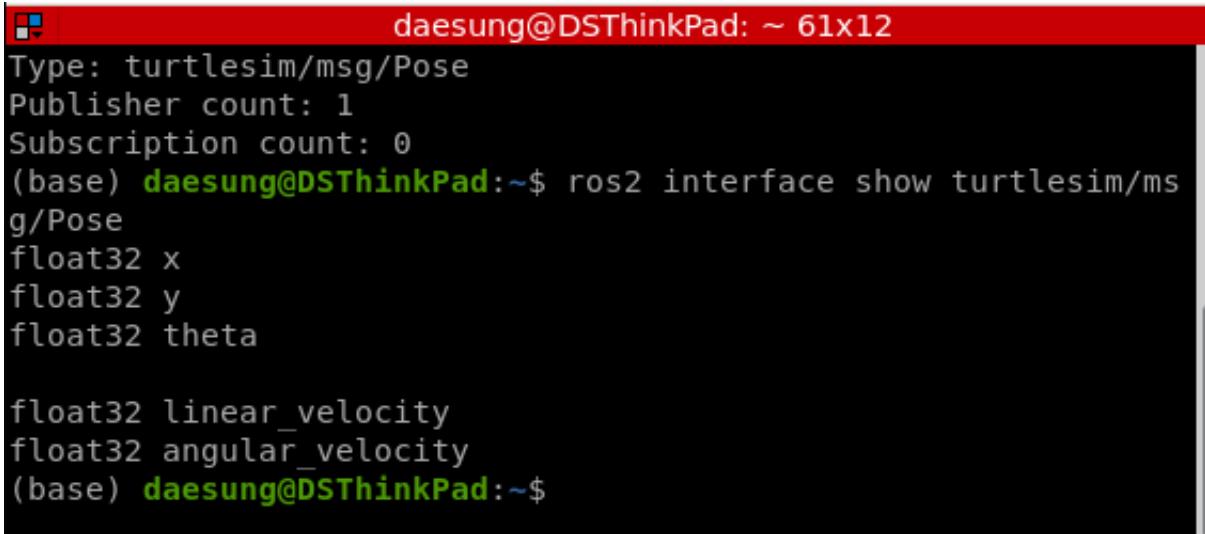
- Topic은 Publish, Subscribe으로 구성되어 있으며 다음과 같이 확인 가능
- ros2 topic info /turtle1/pose

```
daesung@DSThinkPad: ~ 61x12
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
(base) daesung@DSThinkPad:~$ ros2 topic type /turtle1/pose
turtlesim/msg/Pose
(base) daesung@DSThinkPad:~$ ros2 topic info /turtle1/pose
Type: turtlesim/msg/Pose
Publisher count: 1
Subscription count: 0
(base) daesung@DSThinkPad:~$
```

■ Turtlesim 실행

❖ ROS2 Topic 메시지 탑입 확인

- Turtle1이 발행하는 pose Topic 확인을 위해 turtlesim/msg/Pose 확인
- ros2 interface show turtlesim/msg/Pose



```
[daesung@DSThinkPad: ~ 61x12]
Type: turtlesim/msg/Pose
Publisher count: 1
Subscription count: 0
(base) daesung@DSThinkPad:~$ ros2 interface show turtlesim/msg/Pose
float32 x
float32 y
float32 theta

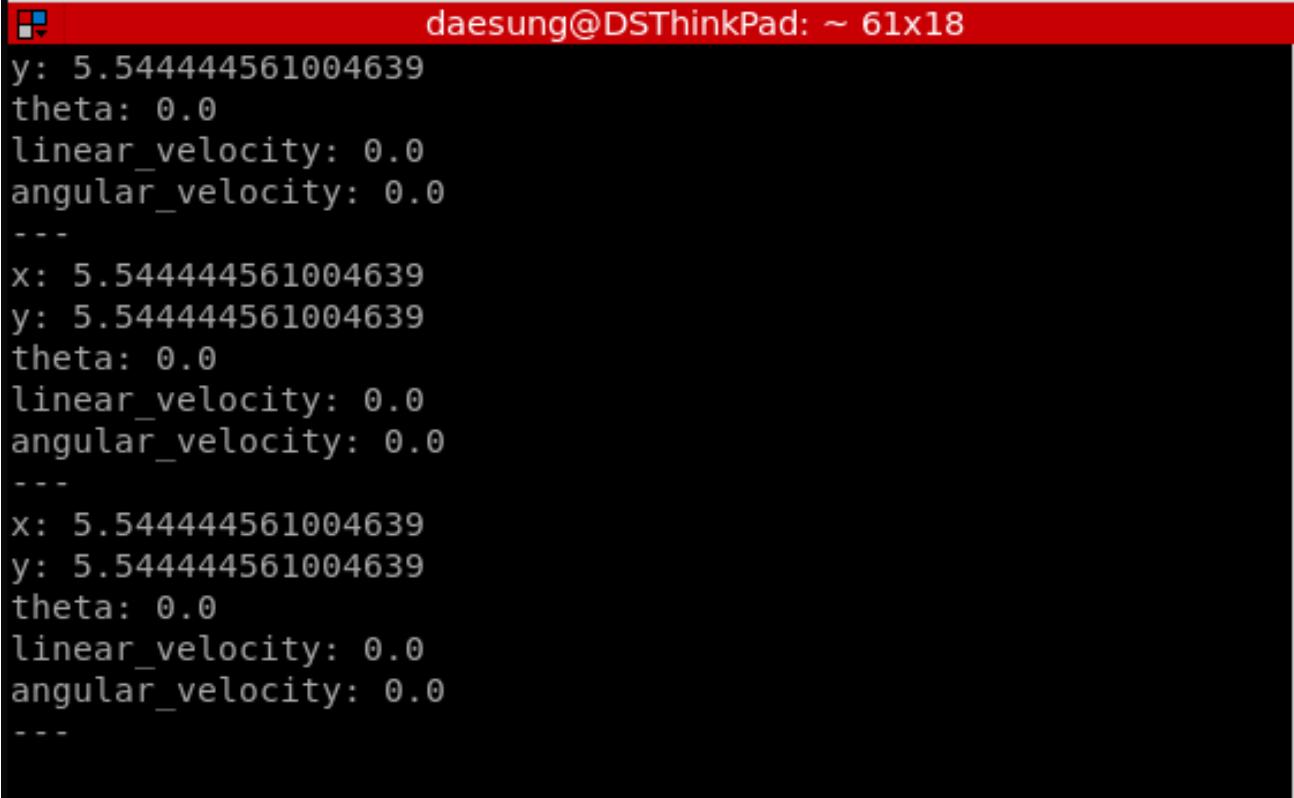
float32 linear_velocity
float32 angular_velocity
(base) daesung@DSThinkPad:~$
```

- Linear_velocity는 직선방향 속도
- Angular_velocity는 회전방향 속도

■ Turtlesim 실행

❖ Terminal에서 Topic Subscribe

- ros2 topic echo /turtle1/pose



A screenshot of a terminal window titled "daesung@DSThinkPad: ~ 61x18". The window contains the output of the command "ros2 topic echo /turtle1/pose". The output shows repeated messages for three turtles, each with fields: x, y, theta, linear_velocity, and angular_velocity. All values are 0.0 except for y which is approximately 5.544444561004639.

```
y: 5.544444561004639
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 5.544444561004639
y: 5.544444561004639
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 5.544444561004639
y: 5.544444561004639
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
```

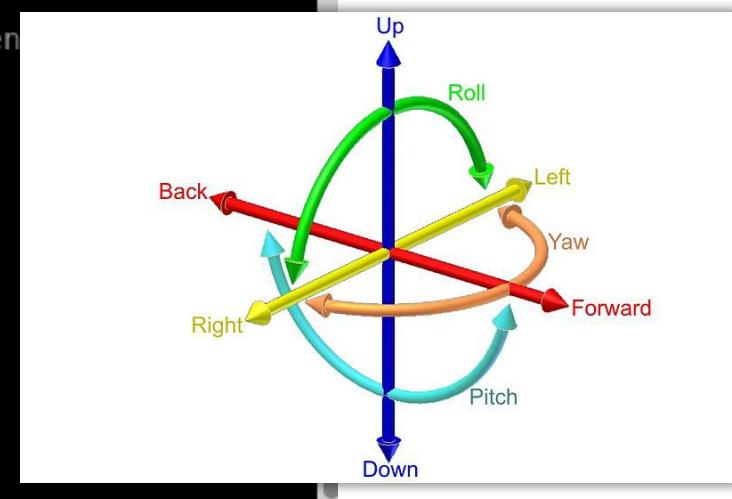
■ Turtlesim 실행

❖ 주행 명령 Topic Publish

- ros2 topic list -t

```
[daesung@DSThinkPad: ~ 61x20]
(base) daesung@DSThinkPad:~$ ros2 topic list -t
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/rosout [rcl_interfaces/msg/Log]
/turtle1/cmd_vel [geometry_msgs/msg/Twist]
/turtle1/color_sensor [turtlesim/msg/Color]
/turtle1/pose [turtlesim/msg/Pose]
(base) daesung@DSThinkPad:~$ ros2 interface show geometry_msg
s/msg/Twist
# This expresses velocity in free space broken
r and angular parts.

Vector3 linear
    float64 x
    float64 y
    float64 z
Vector3 angular
    float64 x
    float64 y
    float64 z
(base) daesung@DSThinkPad:~$
```



■ Turtlesim 실행

❖ 주행 명령 Topic Publish – 직진(한번)

- ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0 }}"

```
(base) daesung@DSThinkPad:~$ ros2 topic pub --once /turtle1/c  
md_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z:  
0.0}, angular: {x: 0.0, y: 0.0, z: 0.0 }}"  
publisher: beginning loop  
publishing #1: geometry_msgs.msg.Twist(linear=ge  
ometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geomet  
ry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
```



■ Turtlesim 실행

❖ 주행 명령 Topic Publish – 회전(한번)

- ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.0 }}"

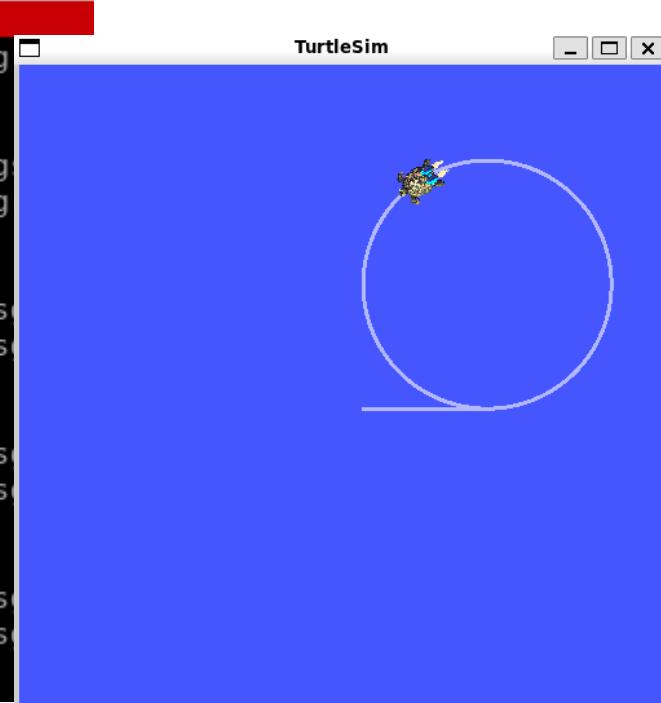
```
(base) daesung@DSThinkPad:~$ ros2 topic pub --once /turtle1/c  
md_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z:  
0.0}, angular: {x: 0.0, y: 0.0, z: 1.0 }}"  
publisher: beginning loop  
publishing #1: geometry_msgs.msg.Twist(linear  
sg.Vector3(x=2.0, y=0.0, z=0.0), angular=geo  
ctor3(x=0.0, y=0.0, z=1.0))
```



■ Turtlesim 실행

❖ 주행 명령 Topic Publish – 주기적 반복

- ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.0 }}"



The screenshot shows a terminal window titled "daesung@DSThinkPad: ~ 61x20" displaying ROS command-line output. The output shows five publishing messages, each defining a Twist message with linear velocity (x=2.0) and angular velocity (z=1.0). To the right of the terminal, the "TurtleSim" application window is open, showing a blue circular track. A small yellow turtle icon is positioned on the track, indicating it is in motion.

```
daesung@DSThinkPad: ~ 61x20
sg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))

publishing #9: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))

publishing #10: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))

publishing #11: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))

publishing #12: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))
```

ROS2 Humble 시작하기

■ Turtlesim 실행

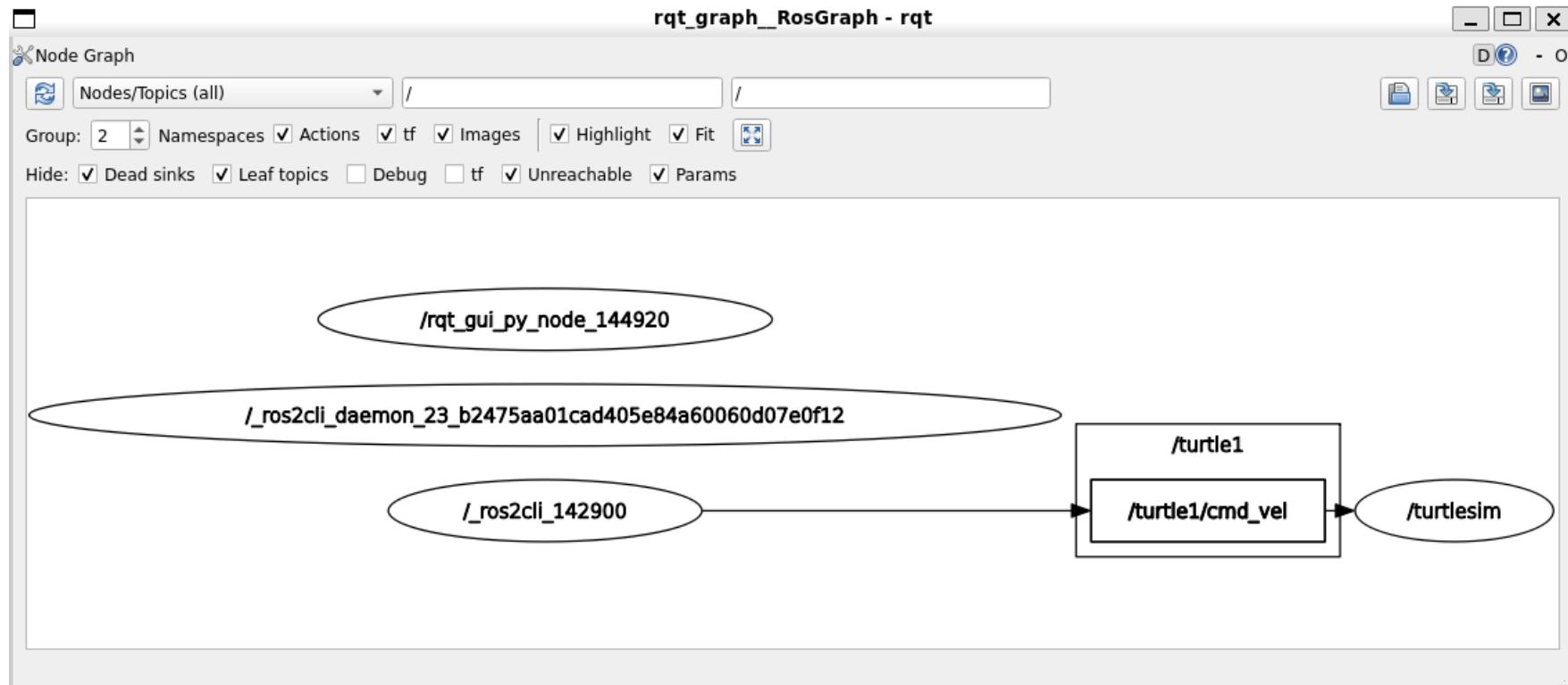
❖ 주행 명령 Topic Flow Check

The screenshot shows three terminal windows on a Linux system (DSThinkPad) with the user daesung.

- Terminal 1:** Shows the command `ros2 run turtlesim turtlesim_node` being run. The output indicates a permission error for the runtime directory and logs the start of the turtlesim node and the spawning of a turtle at coordinates (5.544445, 5.544445) with theta 0.0.
- Terminal 2:** Shows the command `rqt_graph` being run. The output indicates a permission error for the runtime directory.
- Terminal 3:** Shows the command `rqt_publisher` being run. The output shows the publishing of a `geometry_msgs/Twist` message with linear velocity (2.0, 0.0, 0.0) and angular velocity (0.0, 0.0, 1.0).

■ Turtlesim 실행

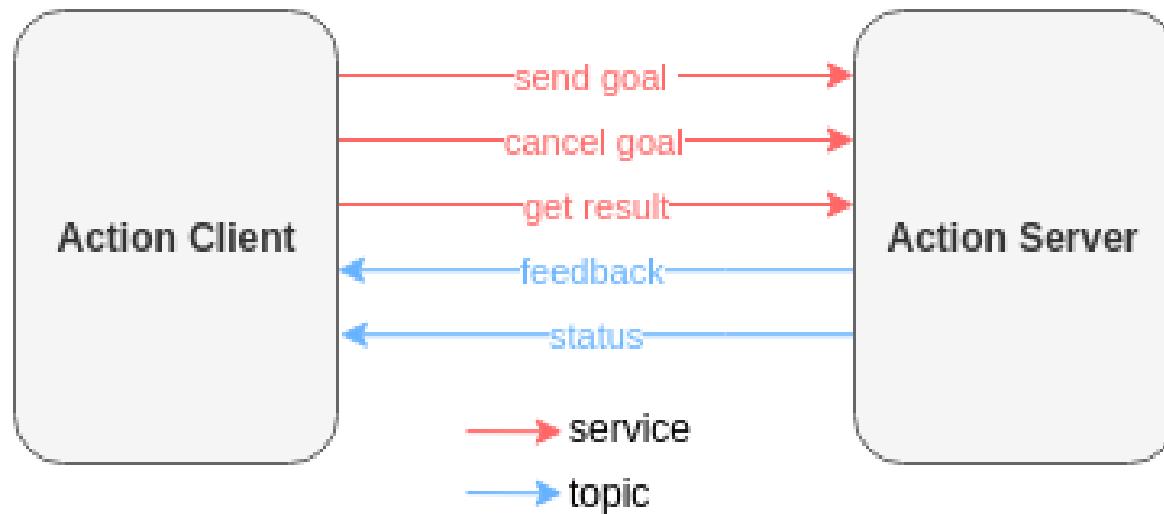
❖ 주행 명령 Topic Flow Check



■ Turtlesim 실행

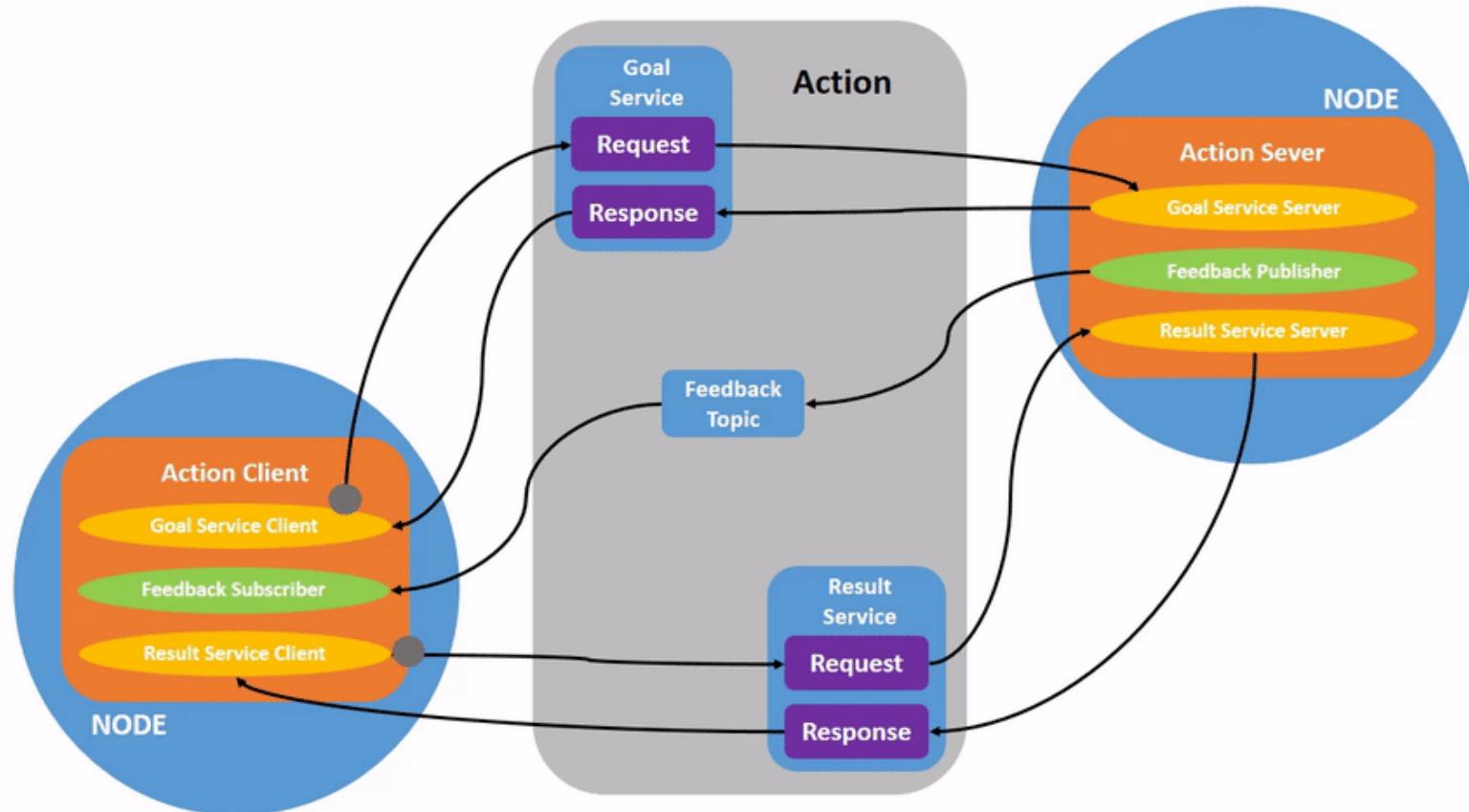
❖ Ros2 Action

- Action은 서비스를 제공하는 Action Server와 Action Client로 구성
- Client에서 목표를 요청하면 목표를 달성할 때 까지 토픽으로 진행 상황에 대한 피드백을 전송
- 목표에 도달하면 결과를 전송



■ Turtlesim 실행

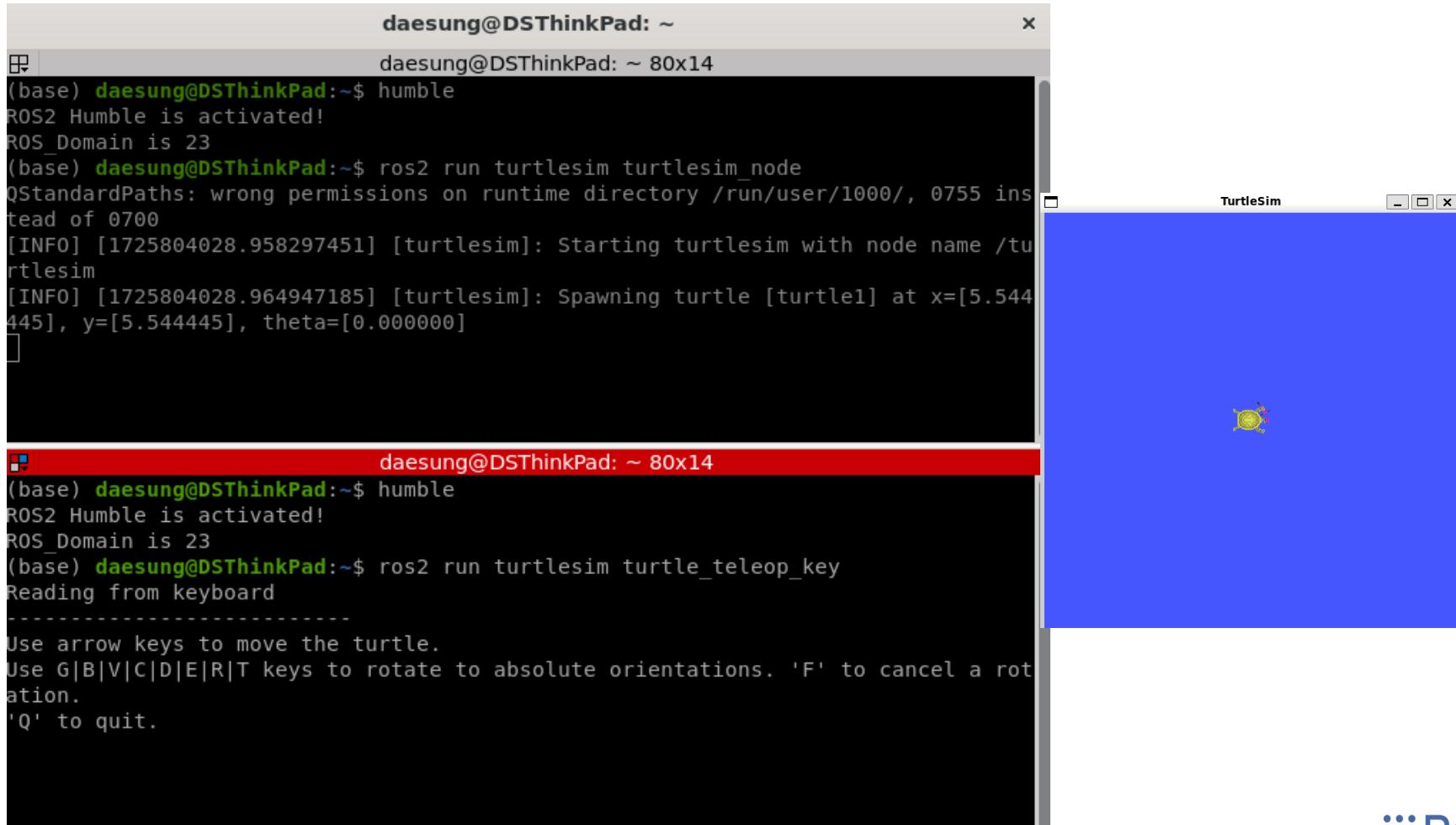
❖ Ros2 Action



ROS2 Humble 시작하기

Turtlesim 실행

Ros2 turtle_teleop_key 노드 실행



```
daesung@DSThinkPad: ~
daesung@DSThinkPad: ~ 80x14
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ ros2 run turtlesim turtlesim_node
QStandardPaths: wrong permissions on runtime directory /run/user/1000/, 0755 instead of 0700
[INFO] [1725804028.958297451] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1725804028.964947185] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
]

daesung@DSThinkPad: ~ 80x14
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ ros2 run turtlesim turtle_teleop_key
Reading from keyboard
-----
Use arrow keys to move the turtle.
Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rotation.
'Q' to quit.
```

ROS2 Humble 시작하기

■ Turtlesim 실행

- ❖ 방향키를 이용해 동작

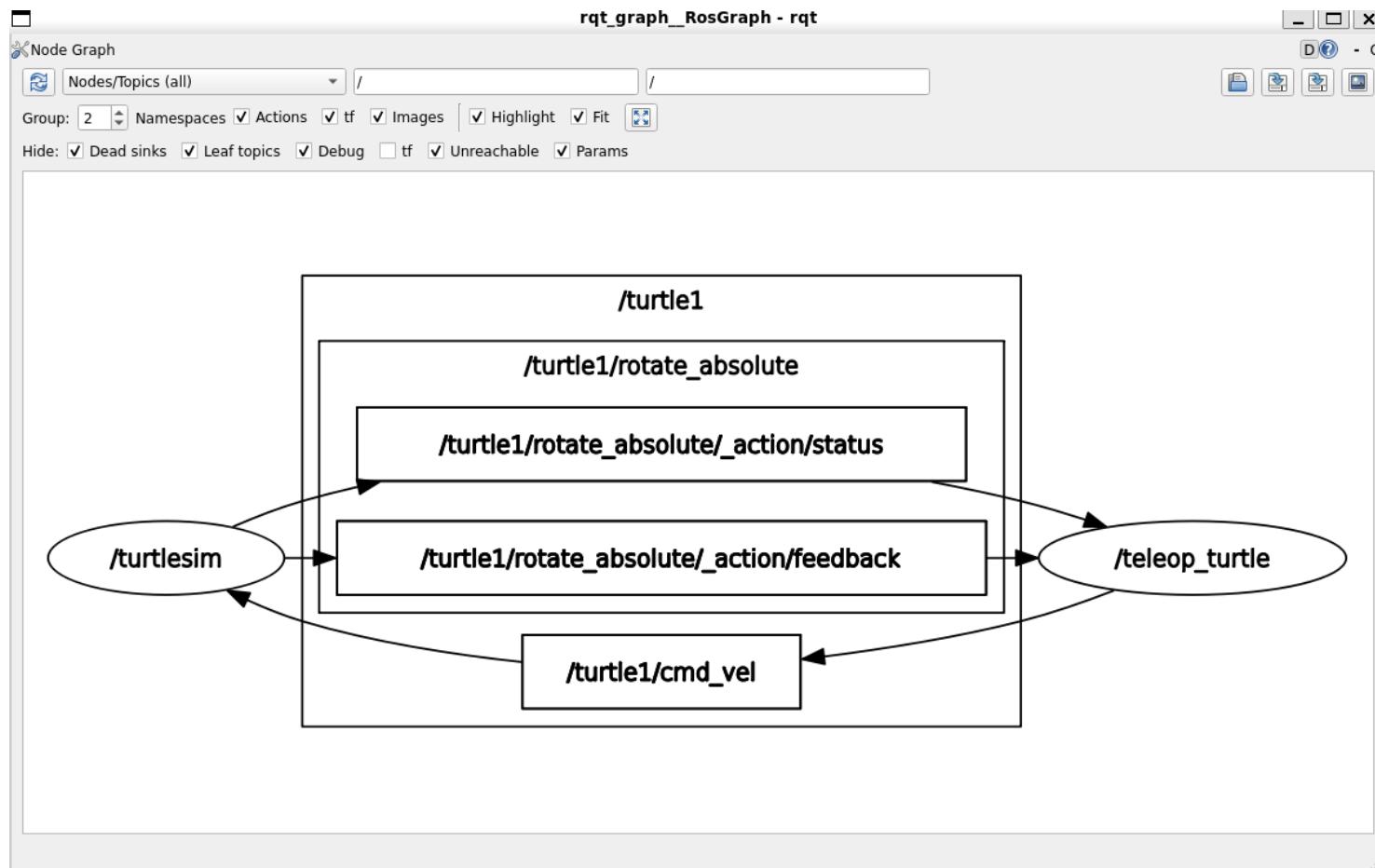
```
daesung@DSThinkPad: ~ 80x14
(base) daesung@DSThinkPad:~$ humble
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ ros2 run turtlesim turtle_teleop_key
Reading from keyboard
-----
Use arrow keys to move the turtle.
Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rotation.
'Q' to quit.

```



■ Turtlesim 실행

- ❖ rqt_graph를 이용한 현재 상태 확인



ROS2 Humble 시작하기

■ Turtlesim 실행

❖ ROS2 action list

```
daesung@DSThinkPad: ~ 38x14
ROS2 Humble is activated!
ROS_Domain is 23
(base) daesung@DSThinkPad:~$ rqt_graph
QStandardPaths: wrong permissions on runtime directory /run/user/1000/, 0755
instead of 0700
(base) daesung@DSThinkPad:~$ ros2 action list
/turtle1/rotate_absolute
(base) daesung@DSThinkPad:~$ ros2 action list -t
/turtle1/rotate_absolute [turtlesim/action/RotateAbsolute]
(base) daesung@DSThinkPad:~$
```

■ Turtlesim 실행

- ❖ ROS2 action interface type

```
daesung@DSThinkPad: ~ 47x16
(base) daesung@DSThinkPad:~$ ros2 action list -t
/turtle1/rotate_absolute [turtlesim/action/RotateAbsolute]
(base) daesung@DSThinkPad:~$ ros2 interface show turtlesim/action/RotateAbsolute
# The desired heading in radians
float32 theta
---
# The angular displacement in radians to the starting position
float32 delta
---
# The remaining rotation in radians
float32 remaining
(base) daesung@DSThinkPad:~$
```

■ Turtlesim 실행

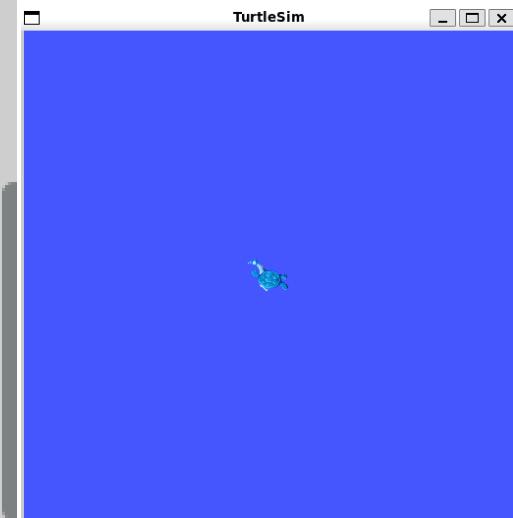
❖ ROS2 action send_goal

```
daesung@DSThinkPad: ~ 47x16
(base) daesung@DSThinkPad:~$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 3.14}"
Waiting for an action server to become available...
Sending goal:
    theta: 3.14

Goal accepted with ID: d4259bc28a9b495eb4e32cc5
0245d6f0

Result:
    delta: -3.135999917984009

Goal finished with status: SUCCEEDED
(base) daesung@DSThinkPad:~$
```





Q & A
