ROS Workspaces

หลังจากที่ทุกๆคนพอจะรู้จักหลักการคร่าวๆของ ROS กันแล้ว ก่อนที่เราจะเริ่มเขียนโปรแกรมได้ เราจะต้องรู้จัก Environment ของการเขียนโปรแกรมก่อนจากที่เรารู้ว่า การที่จะ Program อะไรก็ตามเนี่ย มันจะต้องเขียนโค๊ดก่อน BASED ON C++ and ROSCPP

Code (ที่มนุษย์เข้าใจ .. มั้ง) จะถูกเอาไปแปลงเป็น Binary Code(ที่คอมพิวเตอร์ แต่ละ Architecture เช่น PC, Raspberry pi เข้าใจ) ซึ่งภาษาคน จะเรียกว่า Executable (โปรแกรมที่เราเอาไป Run — ถ้าสมมติเป็น Executableใน Windows จะเป็น file .exe เลย เห็นไม๊) ซึ่งเวลาจะเอาโค๊ดไป แปลง จะเรียกว่า Compile (หรือจะเรียก Build) ก็ได้ เพื่อความเข้าใจ ขอให้คำว่า Compile = Build และจะใช้ คำว่า Compile แทนไปเลยนะคับ

ภาษา Python (ซึ่งเราก็ใช้เขียนCodeใน ROSได้เช่นกัน) ไม่ต้อง Compile นะ — เวลาเราเอา Code ไป ใช้มันจะแปลงเป็น Binary ให้ ตอนเวลาใช้งานเลย (สรุปคือ การ Compile จะใช้กับ C++ ละกันนะ )

ตอนนี้สิ่งที่เรารู้คือจะเขียนโปรแกรม เราจะต้องเขียน Code จะแปลงโปรแกรมเป็นของที่เราสั่งให้ Computer Run ได้ จะต้อง Compile Code ให้อยู่ในรูปของ Executable ซะก่อน

( Beginner tip : ซึ่งใน ภาษา C++ จะต้องตระเตรียมข้อมูลการ Compile ด้วยสิ่งที่เรียกว่า Makefile )

จะแปลงโปรแกรมที่เขียนด้วย ROS ให้ออกมาเป็น Executable แบบ ROS จะต้องใช้ Workspace (Environment ต่างๆ “รวมถึง Code ของเราด้วย” ) ของ ROS ในการสร้างมันเลยมีสิ่งที่เรียกว่า Workspace และ ROS ก็ใช้ระบบที่ Build Workspace ที่เรียกชื่อว่า CATKIN (ของเก่าเรียก ระบบ Build ของใน Workspace ว่า rosbuild)

**Catkin Workspace**

Catkin เป็นระบบ build Code ของ ROS ที่สร้างมาจาก rosbuild (ก็คือระบบเดิมของมันก่อนหน้านี้แหละ) หน้าที่ของมันคือการเก็บ ชุดคำสั่ง(พวก Macro) ที่จะเอาไว้ใช้กับ CMake (คือตัวจัดการ Makefile ) และ จัดการพวก Python script

The name catkin comes from the tail-shaped flower cluster found on willow trees — a reference to Willow Garage where catkin was created.

ระบบ Build Code ทำงานยังไง ? สืบเนื่องจาก ROS จะเก็บ Code เป็น Package (ก็คือ Folder นะแหละ) แล้ว ใน Package จะมี Program ที่เราอยากจะ Compile หรือ build กี่อันก็ได้โดยปกติ Package จะเก็บ Program, Code เป็นเรื่องๆ เช่น Package คุมกล้อง , Package ขับด้วย Joystick อะไรทำนองนี้ ถ้าเป็น ภาษา C++ ตอนเวลาเราเขียนโปรแกรม มันมักจะมี file ที่เป็น ตัว Code และ file ที่เป็น Header (เอาไว้เป็นตัวให้รายละเอียด คร่าวๆ ของ code นั้นๆ) ที่ใช้คู่กัน ทีนี้ ตอนเวลาเราจะ Build หรือ compile มันเนี่ย มันจะต้องมี พิมพ์เขียวของงาน ที่บอกว่า Code อยู่ไหน Header อยู่ไหน และ มันไปใช้ Code ของเจ้าอื่นมาผสมด้วยไหม (เรียกว่าไปใช้ Library) หรือจะบอกว่า Code นี้ จะ build ไปเป็น สถาปัตยกรรม คอมพิวเตอร์อะไร (เช่น จะ build code ไปใส่ computer , build ไปใส่ raspberry pi) พิมพ์เขียวตัวนี้เราจะเรียกว่า Makefile

With CMake, it is specified in a file typically called ‘CMakeLists.txt’ and with GNU Make it is within a file typically called ‘Makefile’. The build system utilizes this information to process and build source code in the appropriate order to generate targets. ROS utilizes a custom build system, catkin, that extends CMake to manage dependencies between packages.

อยากจะใช้งาน Catkin Workspace ละโว้ย เราจะทำอะไรได้บ้าง ? การควบคุม Catkin workspace นั้น เราจะต้องเรียกผ่านสิ่งที่เรียกว่า CLI (Command Line Interface) ซึ่งเป็น Program ที่ต้องเปิดผ่านทาง Terminal

เราจะต้องลง Catkin Build Systemโดยปกติการ Install ROS จะลงของพวกนี้มาให้อยู่แล้ว แต่ถ้าอยากจะลงซ้ำก็

$ sudo apt-get install ros-kinetic-catkin

เราจะต้องสร้าง Catkin Workspace ต่อไปนี้จะใช้คำสั่งพื้นฐานบน Linux หากไม่เข้าใจ อาจต้องไป Search เด้อ เริ่มจากการสร้าง Folder ที่จะเป็น Workspace ในที่นี้ผมจะตั้งชื่อว่า catkin\_ws ซึ่งแม่งจะอยู่ตรงไหนก็ได้ แต่ขอไว้ที่ home ของ Linux ละกัน (Beginner tip : home ของ Linux ก็เหมือน Home ของ Mac และ เอิ่ม . . เหมือน My document ของ Windows ละมั้ง) (ขอ ตั้งชื่อ catkin\_ws เหมือนกับ ROS Tutorial เลยละกัน จริงๆมันจะเป็นชื่ออะไรก็ได้นะ)

$ cd ~

$ mkdir catkin\_ws

เข้าไปสร้าง Folder ที่เก็บ Code ของเรา ชื่อ src (อยู่ใน catkin\_ws อีกที)

(อยู่ใน ~/catkin\_ws มาก่อนแล้วนะ)

$ mkdir src

$ cd src

Initial Workspace (ลงเสาเข็ม เตรียม Site ก่อสร้างก่อน) เข้าไปที่ Folder src แล้วใช้คำสั่ง init workspace ของ Catkin

(อยู่ใน ~/catkin\_ws/src มาก่อนแล้วนะ)

$ catkin\_init\_workspace

จะมีไฟล์ตั้งต้นของ Workspace เกิดขึ้นมา 1 File ชื่อ CMakeLists.txt อย่าไปสนใจมันก่อน มันจะเป็น File ตั้งต้นให้เรา

A screenshot of a cell phone

Description automatically generated

สร้างสิ่งที่จะระบุว่านี่คือ Site ก่อสร้าง Code ของเรา

A screenshot of a computer screen

Description automatically generated

ด้วยการ Build Workspace ที่เปล่าๆ(ยังไม่มี Code 1 ที) ให้เกิด Reference

(เริ่มใน Folder ~/catkin\_ws )

$ catkin\_make

จะพบว่ามี Folder และ file งอกออกมาเต็มเลย นั่นแหละจะเป็นของที่ประกอบขึ้นมาเป็น Workspace ของเรา

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

โดย Folder ที่เกิดขึ้นใน Workspace จะมีดังนี้

src — ไว้เก็บ Code ของเรา \*\*สนใจแค่อันนี้ก่อนนะ

devel—ไว้เก็บตัวที่บอกว่านี่คือ Workspace ของเรา และ ตัวที่ทำการเชื่อม Code เราที่เพิ่งเขียน กับ ของที่เราได้ตอนที่เรา install ROS เข้าด้วยกัน รวมถึงของที่ ROS Generateให้ เช่น custom message ของเราเอง

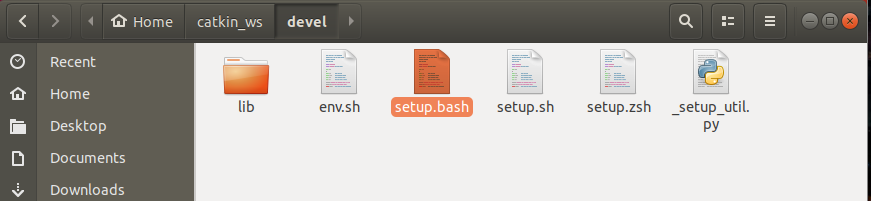
build — Code ที่ถูก Build(compile) เรียบร้อยแล้ว จาก ROS

โดยจริงๆแล้ว เวลาจะใช้งาน จะมีตัวมาเรียก code จากตรงนี้ให้-ไม่ต้องเข้ามา Run เองใน Folder นี้

บอกกับ Linux ว่า ROS Workspace ได้ปักหลักตรงนี้แล้ว

จะได้ประเค็นของที่เราเคย Install ไว้แล้ว (ROS ทั้งหมด รวมถึง Catkin) ให้สามารถมาใช้งานร่วมกับตรงนี้ได้

เรามี ไฟล์ Setup แถมมาให้แล้วว ตอน build !!!!



จะบอกยังไงดี ? ให้ OS รู้จัก ROS Workspace ของเรา ผ่านไฟล์ setup.bash ที่โดนสร้างขึ้นมานี้ จะต้องแก้ตรง bash เลยครับ โดยการบอกไอ้หน้าจอ Terminal ว่า เรามี workspace อยู่ตรงนี้ด้วย Config Bash ที่ไหน ? จะอยู่ใน file ชื่อ .bashrc ภายใต้ Folder home ของเราครับ ใช้ Text editor เปิดมันขึ้นมา (ใช้ Command line Text Editor เท่ๆ ที่ชื่อ nano ละกัน)

$ nano ~/.bashrc

แล้วเติม Code ที่บอก File กรรมสิทธิ์การก่อสร้างของเราต่อท้ายไฟล์เข้าไป

$ source ~/catkin\_ws/devel/setup.bash

กด Save (nano ใช้ Ctrl+X แล้วตอบ y ซึ่งก็คือ yes) หลังจากนี้ “ต้องปิด Terminal และเปิดมันขึ้นมาใหม่”

(Beginner Tips: จริงๆแล้ว เราอยากให้ระบบโหลดไฟล์ที่เพิ่งแก้ใหม่อันนี้แหละ ซึ่งตอนเปิด Terminal มันจะทำให้ — แต่ถ้าอยากทำเอง ใช้คำสั่ง source ~/.bashrc เอาก็ได้)

หลังจากนี้ ระบบ (ทั้ง Linux และ Catkin(ก็คือ ros) ) ก็จะรู้จักว่า workspace เรา Code เราอยู่ตรงนี้แล้ว

หลังจากนี้เวลาเขียน CODE เราจะใช้ คำสั่ง catkin\_make จาก Folder หลักของ workspace (ซึ่งในที่นี้ก็คือ catkin\_ws)

สิ่งที่เรารู้ตอนนี้แบบสั้นๆโง่ๆ

1. เราสร้าง catkin workspace ขึ้นมาได้แล้ว โดย Code จะถูกเก็บไว้ใน Folder “rootของworkspace/src”

2. เวลาจะ build (compile) ให้กลับไปที่ “rootของworkspace” ก่อนแล้วจะเรียกคำสั่ง catkin\_make

3. (แถม) เวลาจะ build แยกทีละ package จะใช้คำสั่ง catkin\_make ชื่อ package ได้ catkin\_make ชื่อpackage ที่ต้องการ Build โดดๆ

คิดว่าตอนนี้เขียนโปรแกรม ROS ได้ละยัง ? ตอบเลยว่า : ยัง 5555555555555+

1.ต้องรู้จัก Package Structure ก่อน

2.ต้องเขียนโครงสร้างของ Makefile ให้เป็นก่อน ! แต่ก่อนที่จะเขียน Makefile ให้เป็น ต้องรู้จักว่าใน Code เราจะต้องมีกี่ไฟล์ มีไฟล์อะไรบ้าง ตั้งชื่อยังไง เก็บยังไงก่อน เพราะบทความที่อ่านมานี้ เราเรียนรู้ ว่าที่ Build Code จะต้องมี Folder อะไรไปแล้ว แต่รอบถัดไปนี้ จะมาดูว่า Code ของเรา จะมีหน้าตาเป็นยังไง