ROS

Module V
Introduction to Robot Operating System

Continuing Education Program
October 26-28, 2018

Dr. Raju Halder
Department of Computer Science & Engineering
IIT Patna

1

References

- Morgan Quigley, Brian Gerkey, and William D. Smart, **Programming Robots with ROS.** O'Reilly Media, First edition.
- Lentin Joseph, Jonathan Cacace , **Mastering ROS for Robotics Programming,** Packt Publishing, Second Revised edition.
- Lentin Joseph, **Learning Robotics Using Python.** Packt Publishing, Second edition.

2

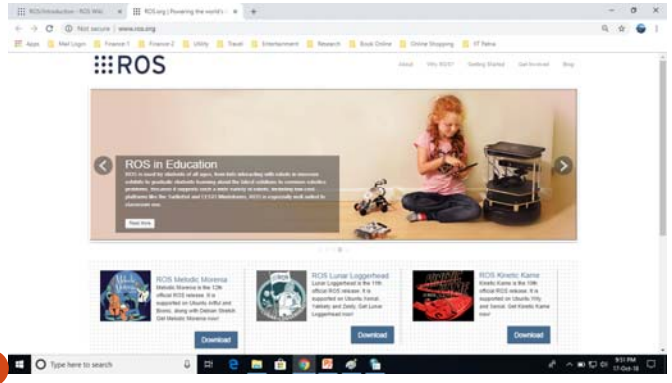
References

- Jason M. O'Kane, **A Gentle Introduction to ROS.** CreateSpace Independent Publishing Platform.
- Wyatt Newman**, A Systematic Approach to Learning Robot Programming with ROS,** Chapman and Hall/CRC, First edition.
- Lentin Joseph , **ROS Robotics Projects,** Packt Publishing, First edition

3

Official Website

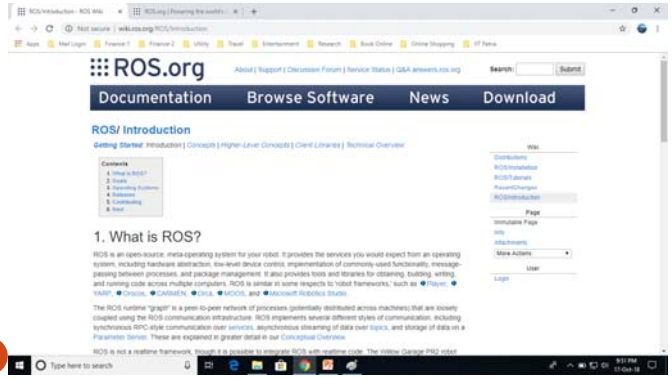
http://www.ros.org **"Powering the world's robots"**



4

Official Website

http://wiki.ros.org/ROS/Introduction



5

Robots using ROS

6

http://wiki.ros.org/Robots

ROS

- ROS: **Robot Operating System**
- It is not exactly same as existing operating systems, like Windows, Linux, etc.
- Flexible framework for writing robot software: Collection of libraries and tools supporting robotic software development.

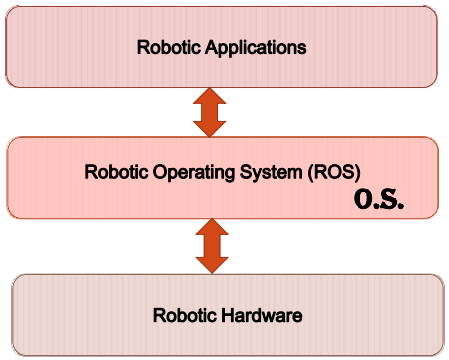
7

ROS

- A suite of user contributed packages that implement common robot functionality such as SLAM, planning, perception, vision, manipulation, etc.
- OS-like functionality: Provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, etc.

8

ROS



9

ROS

"ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers."

Source: "http://wiki.ros.org/ROS/Introduction"

10

History

- Developed at the Stanford Artificial Intelligence Laboratory in 2007.
- Since 2013, ROS is managed by Open Source Robotics Foundation (OSRF).
- De facto standard for robot programming.
- Already used by many robots, popular in both academia and industry.

11

Why ROS

- Code reuse in robotics research and development
- Ready-to-use development environment
- Comprehensive tools and client API libraries (C++, Python, Lisp, Java, …)
- Scalable (distributed network of processes loosely coupled)
- Big community and continuous support: Many device drivers and algorithms are available.

12

Supporting libraries

- openCV: computer vision
- Eigen: Matrix Algebra
- Gazebo: Robot Simulator
- KDL: Kinematics and Dynamics
- TREX: High Level Planning
- PCL: Point Cloud Library
- Many others …

13

ROS Philosophy

- Peer to peer: Individual programs communicate over defined API (ROS messages, services, etc.).
- Distributed: Programs can be run on multiple computers and communicate over the network.
- Multi-lingual: ROS modules can be written in any language for which a client library exists (C++, Python, MATLAB, Java, etc.).
- Light-weight: Stand-alone libraries are wrapped around with a thin ROS layer.
- Free and open-source: Most ROS software is open-source and free to use.

14

ROS =

Plumbing	**Process Management** **Inter-process Communication** **Device Drivers**
Tools	**Simulation** **Visualization** **Graphical User Interface**
Capability	**Control, Planning, Perception** **Mapping, Manipulation**
Ecosystems	**Package Organization** **Software Distribution** **Documentation, Distribution**

15

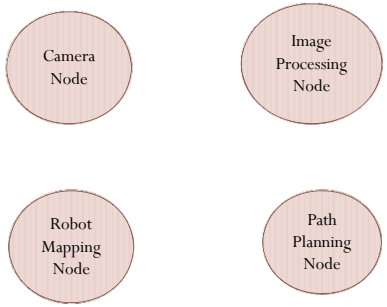
ROS =

Plumbing	**Process Management** **Inter-process Communication** **Device Drivers**
Tools	**Simulation** **Visualization** **Graphical User Interface**
Capability	**Control, Planning, Perception** **Mapping, Manipulation**
Ecosystems	**Package Organization** **Software Distribution** **Documentation, Distribution**

16

Nodes

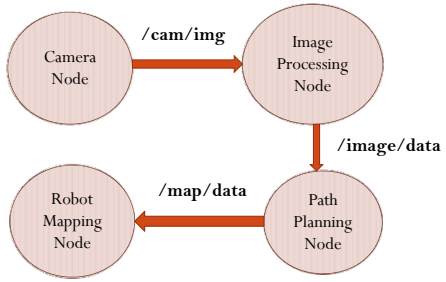
- A node is a process that performs computation.



17

Topics

- Topics are streams of data with publish / subscribe semantics.
- Nodes can publish and subscribe to topic in order to transfer data.

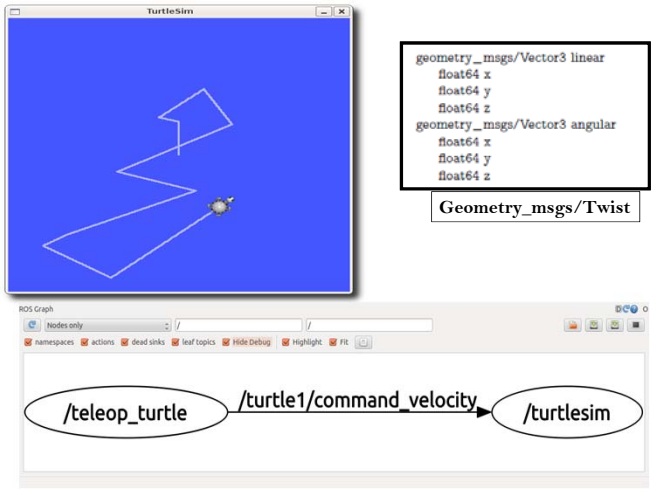


18

Messages

- A message is simply a data structure, comprising typed fields.
- Language agnostic data representation. C++ can talk to Python.
- Messages are sent on defined topics.

19



20

Source: http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv

Built-in field types of ROS messages

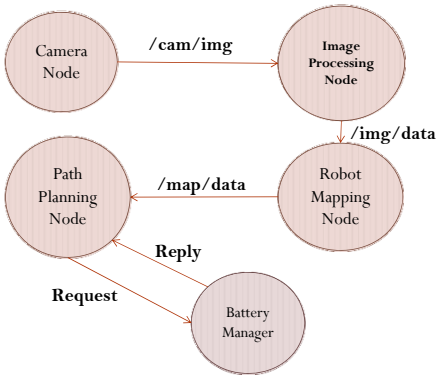
Primitive type	Serialization	C++	Python
bool(1)	unsigned 8-bit int	uint8_t(2)	bool
int8	signed 8-bit int	int8_t	int
uint8	unsigned 8-bit int	uint8_t	int (3)
int16	signed 16-bit int	int16_t	int
uint16	unsigned 16-bit int	uint16_t	int
int32	signed 32-bit int	int32_t	int
uint32	unsigned 32-bit int	uint32_t	int
int64	signed 64-bit int	int64_t	long
uint64	unsigned 64-bit int	uint64_t	long
float32	32-bit IEEE float	float	float
float64	64-bit IEEE float	double	float
string	ascii string(4)	std::string	string
time	secs/nsecs unsigned 32-bit ints	ros::Time	rospy.Time
duration	secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration

21

Source: Mastering ROS for Robotics Programming by Lentin Joseph

Services

- Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply.



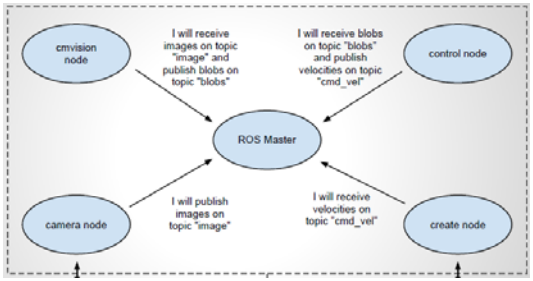
22

ROS Master

- The ROS Master provides name registration and lookup to nodes. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.

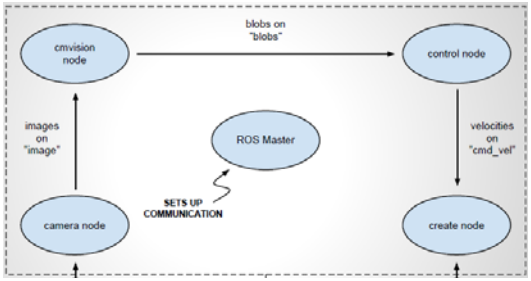
23

ROS Master



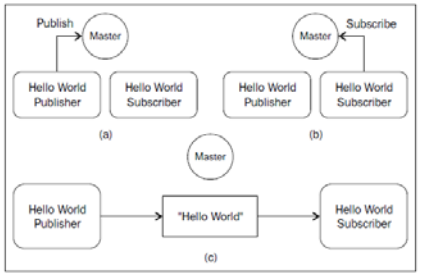
24

ROS Master



25

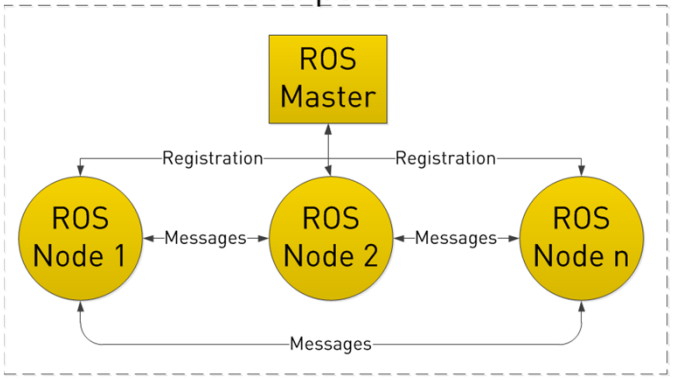
ROS Master



26

Source: Mastering ROS for Robotics Programming by Lentin Joseph

Computer 1

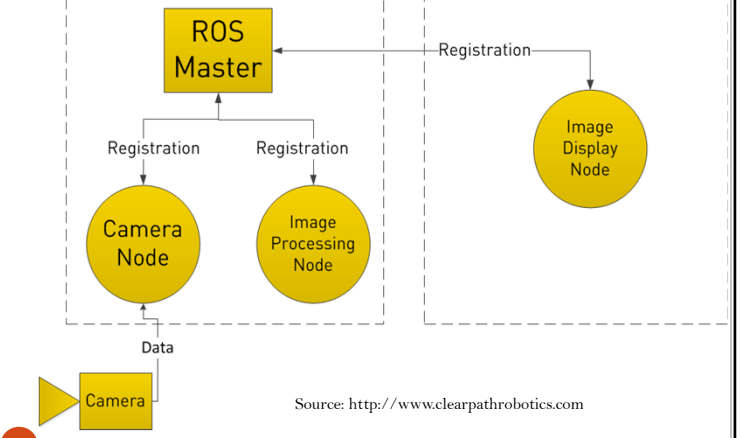


Source: http://www.clearpathrobotics.com

27

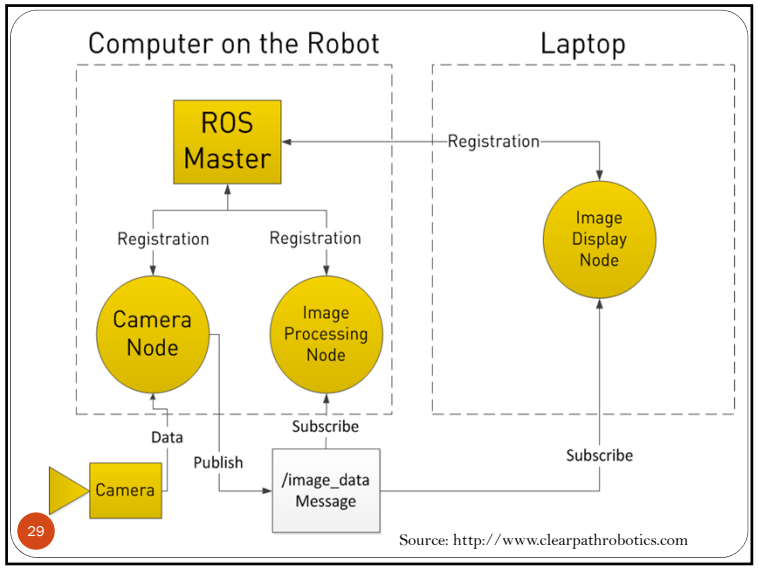
Computer on the Robot

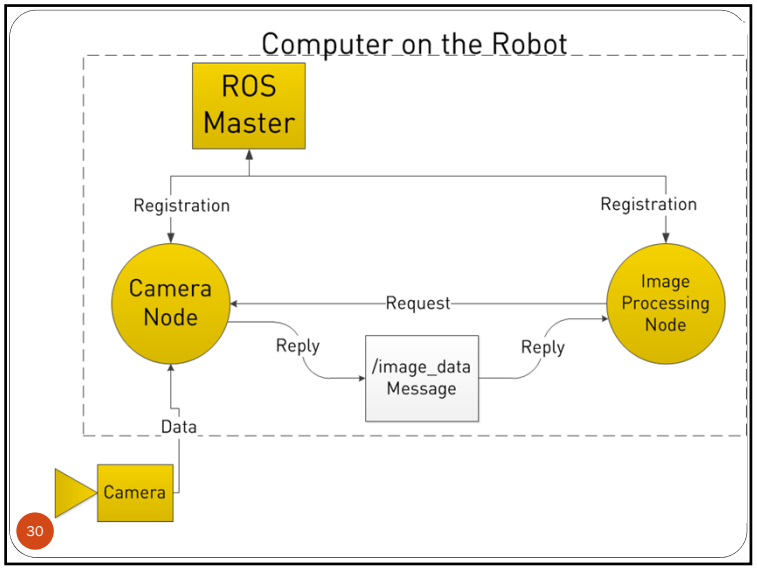
Laptop



Source: http://www.clearpathrobotics.com

28



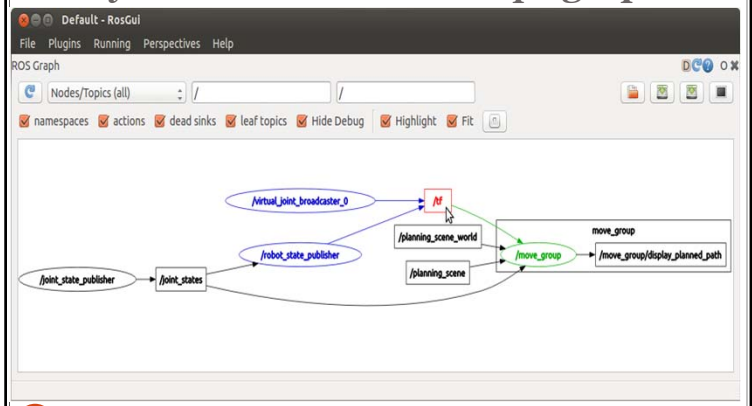


ROS =

Plumbing	**Process Management** **Inter-process Communication** **Device Drivers**
Tools	**Simulation** **Visualization** **Graphical User Interface**
Capability	**Control, Planning, Perception** **Mapping, Manipulation**
Ecosystems	**Package Organization** **Software Distribution** **Documentation, Distribution**

31

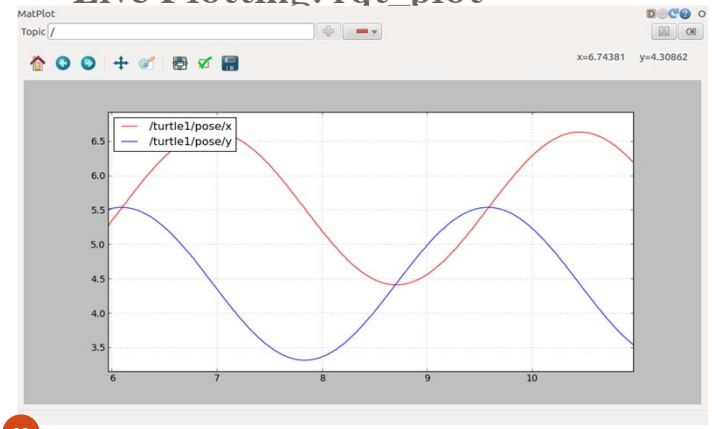
System Visualisation: rqt_graph



32

http://wiki.ros.org/rqt_graph

Live Plotting: rqt_plot

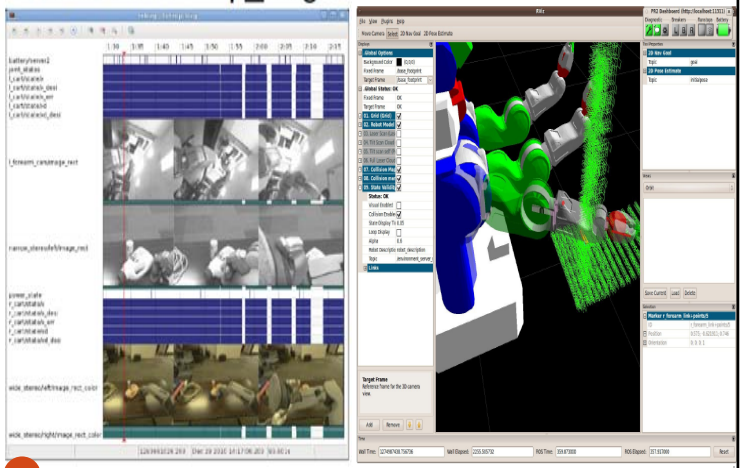


33

http://wiki.ros.org/rqt_plot

ROS tools - rqt_bag

ROS tools - rviz



34

http://wiki.ros.org/Tools

ROS =

Plumbing	Process Management Inter-process Communication Device Drivers
Tools	Simulation Visualization Graphical User Interface
Capability	Control, Planning, Perception Mapping, Manipulation
Ecosystems	Package Organization Software Distribution Documentation, Distribution

35

ROS =

Plumbing	Process Management Inter-process Communication Device Drivers
Tools	Simulation Visualization Graphical User Interface
Capability	Control, Planning, Perception Mapping, Manipulation
Ecosystems	Package Organization Software Distribution Documentation, Distribution

36

ROS Statistics July 2016 - July 2017

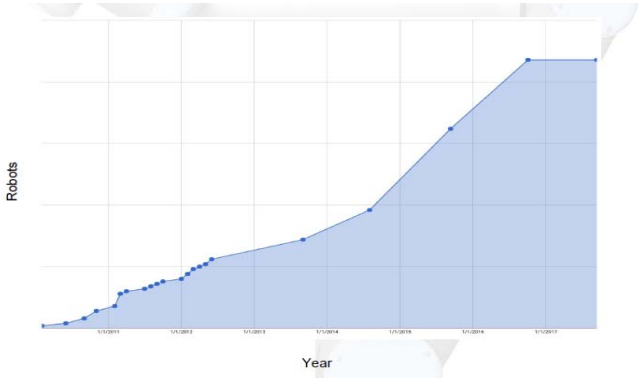
- **Total traffic on packages.ros.org:**
  - 232,577 Unique Visitors (105 % increase)
  - 4714.22 GB (54% increase)
- **Total downloads of .deb packages:**
  - 13,441,711 (59% increase)
- **Unique package names downloaded as .deb files:**
  - 9395 (24% increase)
- **Number of unique versions of .deb packages downloaded:**
  - 53,382 (16 % decrease)

37

Worldwide User Base

38

No. of Robots Supporting ROS



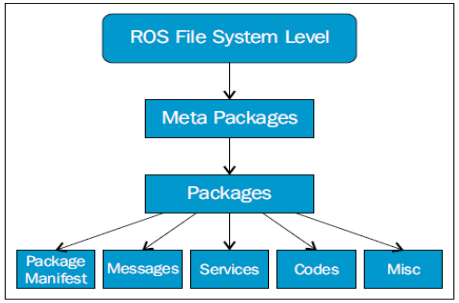
39

ROS Packages

- Packages are the main unit for organizing large complex software in ROS.
- Packages are the most atomic build item and release item in ROS.
- A package may contain ROS source code, launch files, configuration files, message definitions, datasets, or anything else that is usefully organized together

40

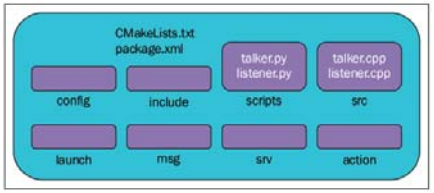
ROS file system level



41

Source: Mastering ROS for Robotics Programming by Lentin Joseph

ROS Packages



42

Source: Mastering ROS for Robotics Programming by Lentin Joseph

ROS Packages

- **config:** Consists of all configuration files.
- **include/package_name:** This folder consists of headers and libraries required in the package.
- **scripts:** This folder keeps executable Python scripts.
- **src:** This folder stores the C++ source codes.
- **launch:** This folder keeps the launch files that are used to launch one or more ROS nodes.
- **msg:** This folder contains custom message definitions.
- **srv:** This folder contains the service definitions.
- **action:** This folder contains the action definition.
- **package.xml:** This is the package manifest file of this package.
- **CMakeLists.txt:** This is the CMake build file of this package.

43

ROS Packages: A Sample Package

```
mastering_ros_demo_pkg/
|-- action
|   `-- Demo_action.action
|-- CMakeLists.txt
|-- include
|-- msg
|   `-- demo_msg.msg
|-- package.xml
|-- src
|   |-- demo_action_client.cpp
|   |-- demo_action_server.cpp
|   |-- demo_msg_publisher.cpp
|   |-- demo_msg_subscriber.cpp
|   |-- demo_service_client.cpp
|   |-- demo_service_server.cpp
|   |-- demo_topic_publisher.cpp
|   `-- demo_topic_subscriber.cpp
`-- srv
    `-- demo_srv.srv
```

44

Source: Mastering ROS for Robotics Programming by Lentin Joseph

ROS Packages: package.xml

```xml
<?xml version="1.0"?>
<package>
  <name>hello_world</name>
  <version>0.0.0</version>
  <description>The hello_world package</description>

  <maintainer email="qboticslabs@gmail.com">Lentin Joseph</maintainer>
  <license>BSD</license>
  <url type="website">http://wiki.ros.org/hello_world</url>
 <author email="qboticslabs@gmail.com">Lentin Joseph</author>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>

  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>

  <export>
  </export>
</package>
```

45

Source: Mastering ROS for Robotics Programming by Lentin Joseph

ROS Packages: CMakeLists.txt

Required CMake Version (cmake_minimum_required)

Package Name (project())

Find other CMake/Catkin packages needed for build (find_package())

Enable Python module support (catkin_python_setup())

Message/Service/Action Generators (add_message_files(), add_service_files(), add_action_files())

Invoke message/service/action generation (generate_messages())

Specify package build info export (catkin_package())

Libraries/Executables to build (add_library()/add_executable()/target_link_libraries())

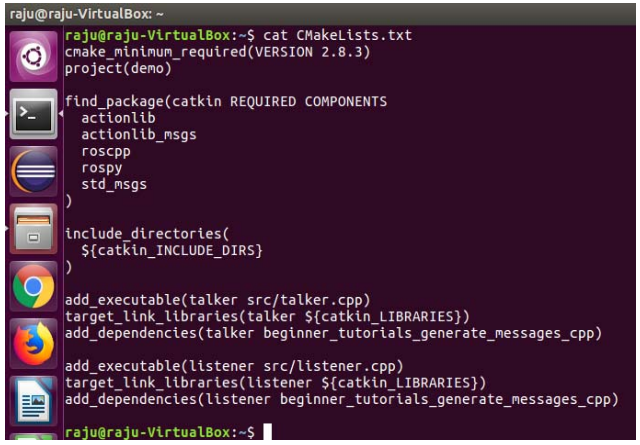
Tests to build (catkin_add_gtest())

Install rules (install())

46

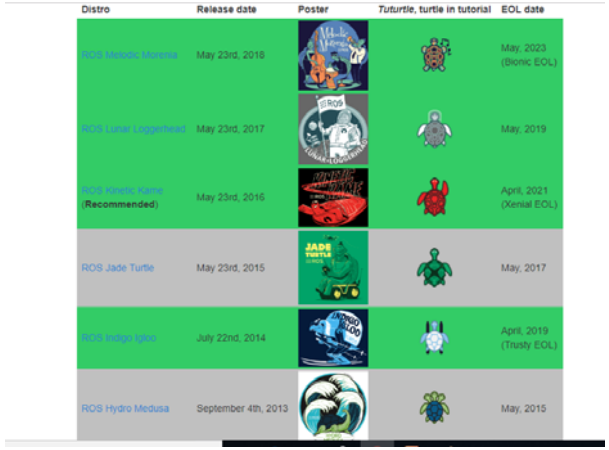
Source: http://wiki.ros.org/catkin/CMakeLists.txt

ROS Packages: CMakeLists.txt



47

ROS Distribution Releases



48

ROS Supported Platforms

- ROS is currently supported only on Ubuntu
  - Other variants (Windows and Mac OS X) are considered experimental
- ROS Kinetic supports the following Ubuntu versions:
  - Wily (Ubuntu 15.10),
  - Xenial (Ubuntu 16.04),
  - Jessie (Debian 8)

49

Basic Unix Command

- **ls:** Lists files and folders in the working directory
- **cd <folder>:** Change the working directory to **<folder>**
- **pwd:** Prints the current working directory
- **mkdir <directory>:** Creates a directory in your working directory named **<directory>**
- **mv <src> <dest>:** Move files from **<src>** to **<dest>**
- **cp <src> <dest>:** Copies files from **<src>** to **<dest>**
- **gedit or nano <file>:** Opens a text editor to edit **<file>**
- **sudo apt-get install ros-kinetic-desktop:** Install as a root users using Advanced Package Tool.

50

ROS Installation

- If you already have Ubuntu installed, follow the instructions at:
  http://wiki.ros.org/Kinetic/Installation/Ubuntu
- Also you can use Oracle VM VirtualBox
  https://www.virtualbox.org
- You can also download a VM with ROS Pre-installed from here:
  http://nootrix.com/downloads/#RosVM

51

Installing ROS

- Set up your system to accept ROS packages from packages.ros.org.

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

- Add apt-keys. The apt-key is used to manage the list of keys used by apt to authenticate the packages.

```
$ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116
```

- Update the Ubuntu package index.

```
$ sudo apt-get update
```

- Install the packages.

```
$ sudo apt-get install ros-kinetic-desktop-full
```

52

Installing ROS (Cont…)

- Initializing ROSDEP

```
$ sudo rosdep init
$ rosdep update
```

- Add ROS environmental variables to the .bashrc file.

```
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

- Add…

```
$ sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

53

Setting ROS Workspace

- Every new project should be organized in packages
- We need to work in a very specific ROS workspace, which is known as the catkin workspace. (catkin_ws)
- CREATING ROS WORKSPACE (Only one time)
  - The default directory for ROS packages is the path: /opt/ros/kinetic/share/
  - Verify it with the command: printenv| grep ROS
  - Directory path for new ROS projects: ~/catkin_ws/src

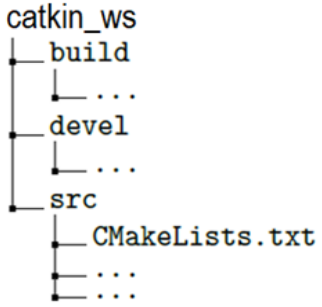
54

Setting ROS Workspace

- **Create an empty workspace and a folder 'src' in it to store packages:**
  - mkdir -p ~/catkin_ws/src
- **GO to 'src' and initialize a catkin workspace**
  - cd ~/catkin_ws/src
  - catkin_init_workspace
- **Go into your empty workspace and build packages**
  - cd ~/catkin_ws/
  - catkin_make
- **Before continuing make sure you source your new setup.*sh file**
  - source devel/setup.bash
  - echo $ROS_PACKAGE_PATH
- **Test your path:** /home/youruser/catkin_ws/src:/opt/ros/kinetic/share

55

Setting ROS Workspace: An Example



56

Integrate Eclipse with ROS: Install Eclipse

- sudo apt-get install default-jre
- Download Eclipse C/C++ IDE from: https://www.eclipse.org/downloads/packages/
- tar -xfvz <downloadedfile>
- sudo mv eclipse /opt
- cd /opt
- sudo chmod -R 777 eclipse
- sudo ln -s /opt/eclipse/eclipse /usr/bin/eclipse

57

Install Eclipse

- $sudo gedit /usr/share/applications/eclipse.desktop

```
[Desktop Entry]
Name=Eclipse
Type=Application
Exec=bash -i -c "/opt/eclipse/eclipse"
Terminal=false
Icon=/opt/eclipse/icon.xpm
Comment=Integrated Development Environment
NoDisplay=false
Categories=Development;IDE
Name[en]=eclipse.desktop
```

The bash –i - c command will cause your IDE's launcher icon to load your ROS-sourced shell environment before launching eclipse

58

Install Eclipse

- Alternate Option:

$ sudo apt-get install eclipse eclipse-cdt g++

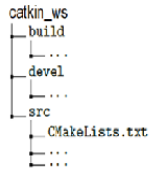
59

Make Eclipse Project Files

- Go to workspace directory and run catkin_make with options to generate eclipse project files:

```
$ cd ~/catkin_ws
$ catkin_make --force-cmake -G"Eclipse CDT4 - Unix Makefiles"
```

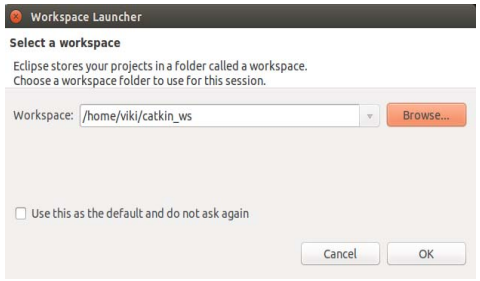
- The project files will be generated in ~/catkin_ws/build.



60

Import the Project into Eclipse

- Now start Eclipse
- Choose catkin_ws folder as the workspace folder



61

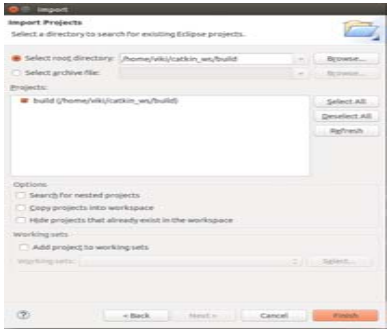
Import the Project into Eclipse

- Choose File --> Import --> General --> Existing Projects into Workspace

62

Import the Project into Eclipse

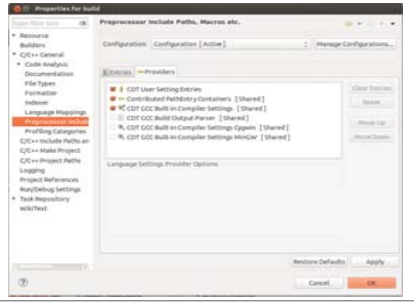
- Now import the project from the ~/catkin_ws/build folder



63

Fix Preprocessor Include Paths

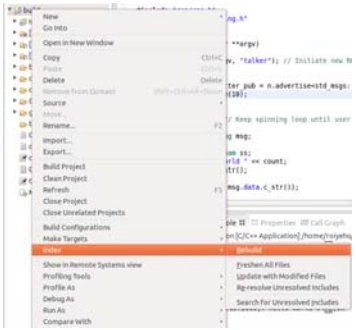
- By default, the intelligence in Eclipse won't recognize the system header files (like <string>). To fix that:
  - Go to Project → Properties --> C/C++ General --> Preprocessor Include Paths, Macros, etc. --> Providers tab
  - Check CDT GCC Built-in Compiler Settings [Shared]



64

Fix Preprocessor Include Paths

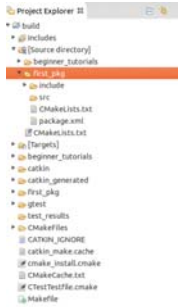
- After that rebuild the C/C++ index by Right click on project -> Index -> Rebuild



65

Project Structure

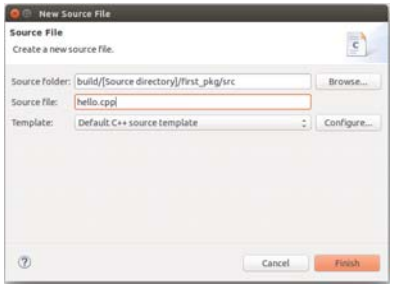
- Eclipse provides a link "Source directory" within the project so that you can edit the source code



66

Add New Source File

- Right click on src and select New –> Source File, and create a file named hello.cpp



67

Code Completion

- Use Eclipse standard shortcuts to get code completion (i.e., Ctrl+Space)

68

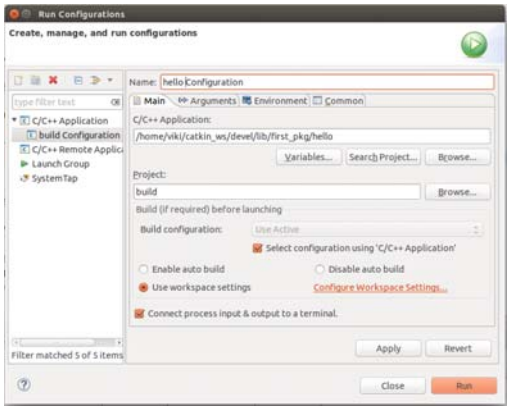
Running the Node Inside Eclipse

- Click on: Project → build project
- Create a new launch configuration, by clicking on Run → Run configurations… → C/C++ Application (double click or click on New).
- Select the correct binary on the main tab (use the Browse… button)

  ~/catkin_ws/devel/lib/first_pkg/hello
- Make sure roscore is running in a terminal
- Click Run

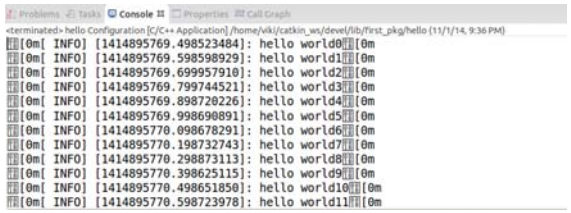
69

Running the Node Inside Eclipse



70

Running the Node Inside Eclipse



71

Debugging

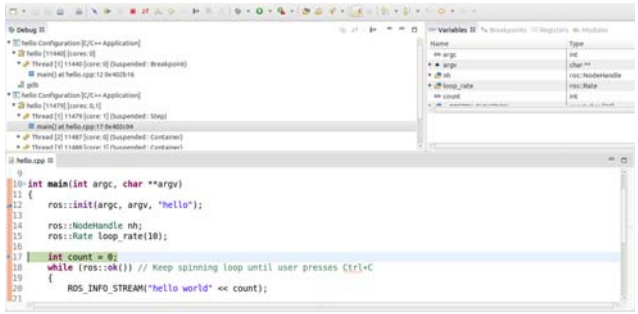
- To enable debugging, you should first execute the following command in catkin_ws/build:

```
$ cmake ../src -DCMAKE_BUILD_TYPE=Debug
```

- Restart Eclipse
- Then you will be able to use the standard debugging tools in Eclipse

72

Debugging



73