ECE 478-578
Intelligent Robotics I

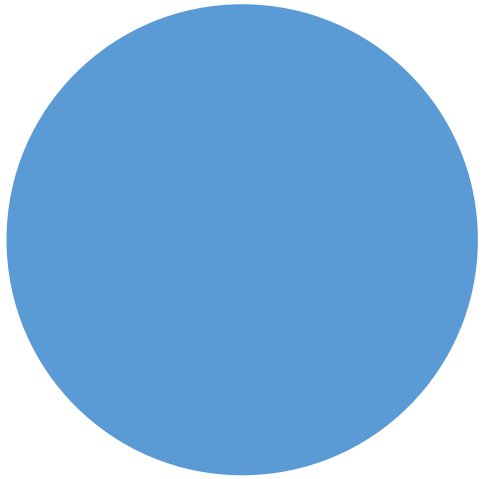
PhD. Husnu Melih Erdogan – Electrical & Computer Engineering

herdogan@pdx.edu **Teaching Assistant**

Portland State
UNIVERSITY

Introduction to ROS Part - 2

Portland State
UNIVERSITY



ROS
(Robot Operating System)

Portland State
UNIVERSITY

Course Structure

- **Part 1 - Overview**
  - What is ROS?
  - Introduction to ROS
  - ROS architecture, philosophy, history
  - How to install ROS?
  - Examples
  - Installation
  - ROS Master
  - ROS Nodes
  - ROS Topic
  - ROS Messages
  - Console Commands
  - ROS Packages
  - ROS Launch-files
  - Catkin Workspace and Build System
  - Turtlesim

  - Assignment 3 – Part1

- **Part 2 - Details**
  - ROS File System
  - ROS Package
  - How to create a package?
  - How to build a package?
  - Creating a Publisher Node
  - Creating a Subscriber Node
  - Creating a Launch File

  - Assignment 3- Part2

- **Part 3 - Details**
  - How to use ROS .bagfiles?
  - ROS Parameters
  - ROS Messages
  - ROS Services
  - ROS Actions

  - Assignment 4

- **Part 4 - Project**
  - Rviz
  - Mapping
  - Localization
  - 2D Multi-Robot Simulator

  - Assignment 5

Portland State UNIVERSITY

ROS
(Robot Operating System Review)

Portland State
UNIVERSITY

ROS Workspace Environment

- Standard name for it **catkin_ws**
- However, you can use whatever name you want.
- You can create it in any user location

- Before you use a work space first you need to source your setup.*sh.
- The setup.bash files setup your environment variables and paths.

Portland State
UNIVERSITY

ROS Workspace Environment

- setup.bash Environment Setup File
- The binary catkin package includes a set of environment setup files that are used to extend your shell environment, so that your terminal can find and use any resources that have been installed to that location.
- the setup file included in the root of the distribution install directory (usually /opt/ros/<ros_distribution_name>)
- For example with ROS Kinetic on Ubuntu the files would be located as such:

```
/
  opt/
    ros/
      kinetic/
        setup.bash -- Environment setup file for Bash shell
        setup.sh   -- Environment setup file for Bourne shell
        setup.zsh  -- Environment setup file for zshell
        ...
```

Portland State UNIVERSITY

ROS Workspace Environment

- If you want to change the environment of your current shell
  - **source /opt/ros/kinetic/setup.bash**

- If you want to overlay a workspace on top of your environment.
  - **source devel/setup.bash**

- If you want to source your terminal automatically.
  - **echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc**
  - **source ~/.bashrc**

Portland State UNIVERSITY

Catkin Workspace

What is catkin?

A catkin workspace is a folder where you modify, build, and install catkin packages. (wiki.ros.org)

Catkin packages can be built as a standalone project, in the same way that normal cmake projects can be built, but ***catkin also provides the concept of workspaces, where you can build multiple, interdependent packages together all at once.***

Create a Catkin Workspace

- **mkdir -p ~/catkin_ws/src**

(create a new directory called src in another directory called catkin ws)

- **cd ~/catkin_ws/src**

(go into the src directory)

- **catkin_init_workspace**

(initialize the workspace)

- **cd ~/catkin_ws/**

(go into your brand new workspace)

- **catkin_make**

(build your code in a catkin workspace)

- **source devel/setup.bash**

(Before continuing make sure you source your new setup.*sh file)

- **echo $ROS_PACKAGE_PATH**

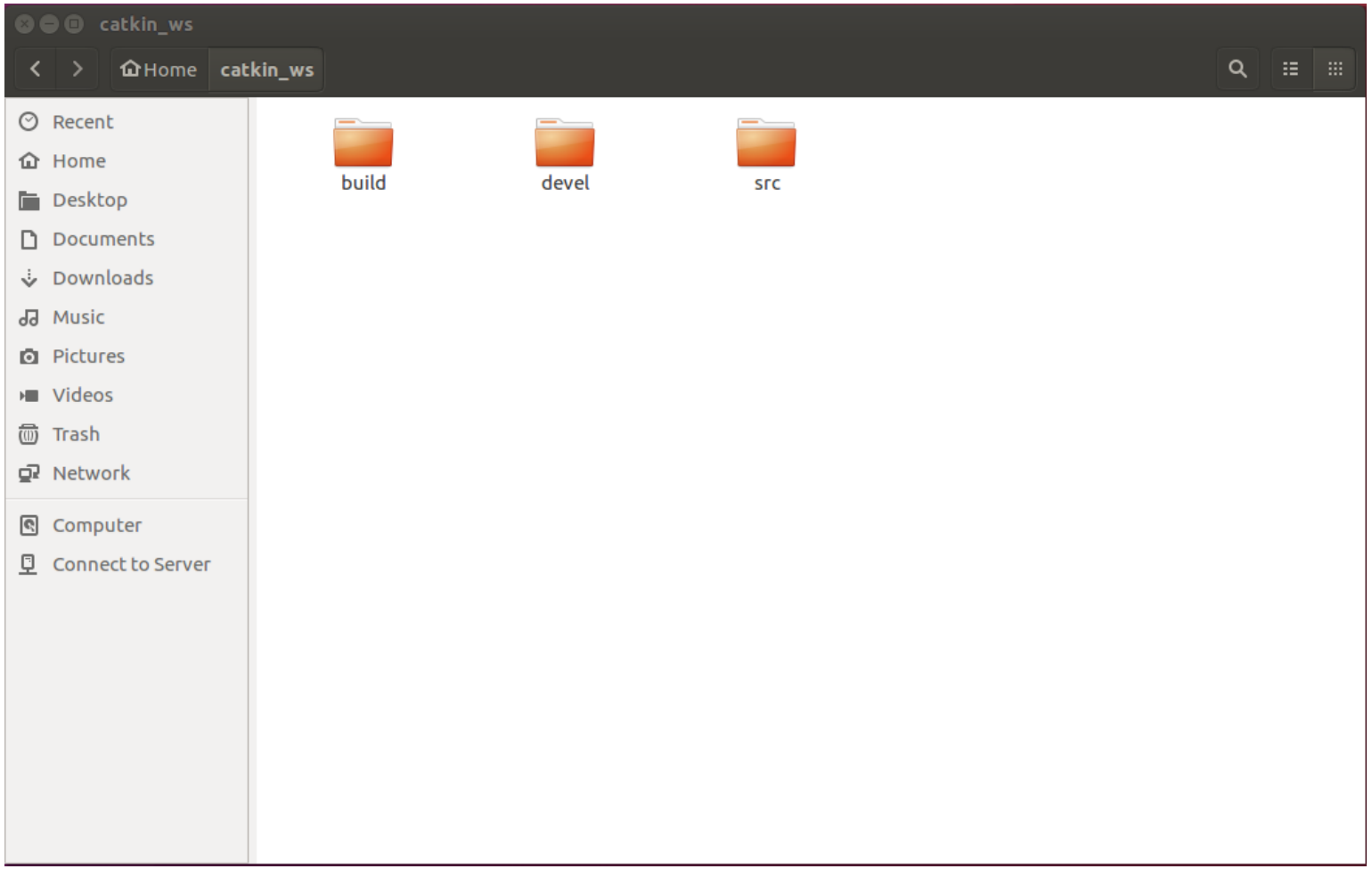
Test your path: /home/youruser/catkin_ws/src:/opt/ros/kinetic/share

Catkin Workspace

- ***Do not use catkin_make and catkin build at the same time***
- ***Do not mix them***

Portland State UNIVERSITY

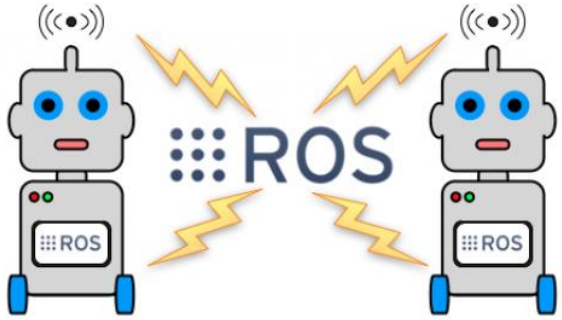
Catkin Workspace



Portland State
UNIVERSITY

ROS Master

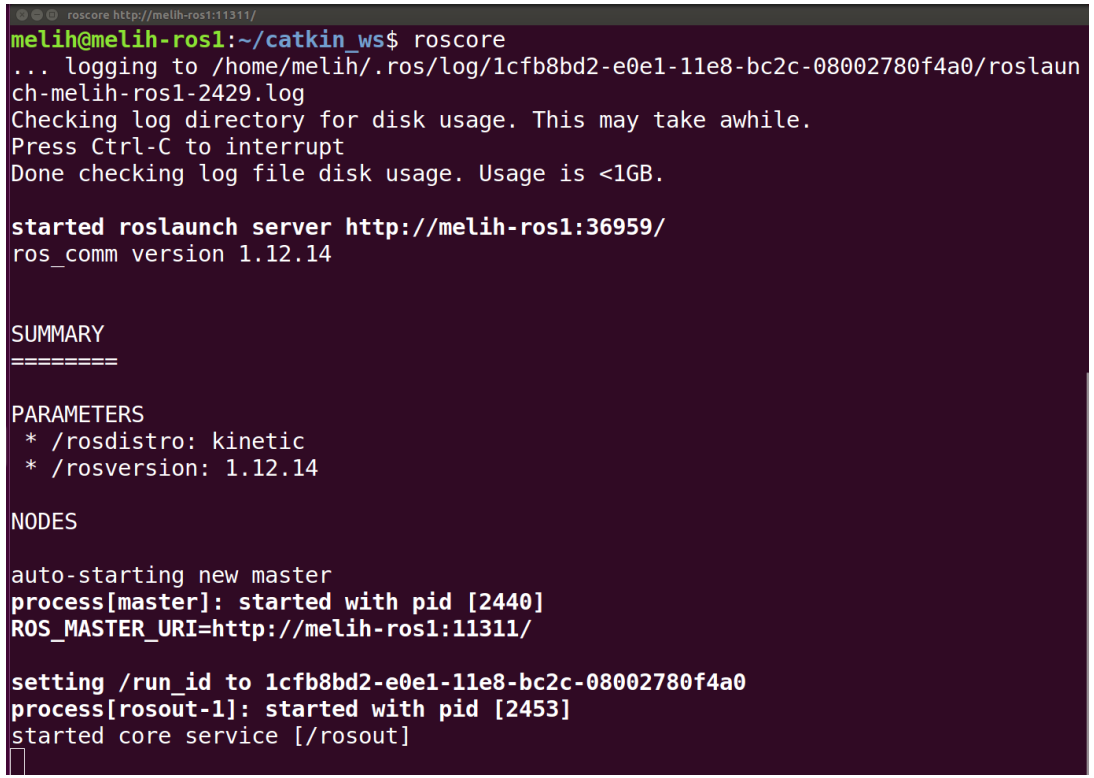
- In charge of managing the communication between nodes
- Every node registers with the master when you run them
- It tracks publishers and subscribers to topics as well as services.
- The role of the Master is to enable individual ROS nodes to locate one another.
- Once these nodes have located each other they communicate with each other peer-to-peer.
- The Master also provides the parameter server.



Portland State UNIVERSITY

ROS Master

- In order to start a master type:
  - **roscore**



Portland State UNIVERSITY

Ros Nodes

- Single-purpose, executable program written in Python, C++, Matlab etc
- Individually complied, executed and managed.
- You can run, stop, and kill nodes whenever you want.
- They run in parallel.
- In order to run a node type:
  - **rosrun package_name node_name**
- See active nodes running on ROS
  - **rosnode list**
- In order to get more information about nodes type:
  - **rosnode info node_name**

Portland State UNIVERSITY

ROS Topics

- ROS Nodes communicate over topics.
- Each node can **publish** or **subscribe** topics
- When one of the nodes register with the master, node also tells master that I am the publisher of these topics. Therefore, other nodes can know who publishes that topic and subtribe that topic.
- Topics is used to transfer data from publisher node to subscriber node
- List all active topics :
  - **rostopic list**
- Subscribe and print the contents of a topic
  - **rostopic echo /topic-name**
- Show information about the topic
  - **rostopic info /topic-name**
- Show all the other possible rostopic commands
  - **rostopic -h**

Portland State UNIVERSITY

ROS Messages

- ROS message describes the data values that ROS nodes publish.
- Messages help ROS tolls to automatically generate code for the message type in several target languages.
- Message descriptions are in .msg files in the msg/ directory.
- Messages can be nested in each other.
- Each ROS distribution can have a different description for a message
  - You can go to http://wiki.ros.org/msg and check the message type
- Publish a message to a subscriber ROS node on command line.
  - rostopic pub /mytopic std_msg/String “data: ‘Portland State University’”

Summary of ROS Messages

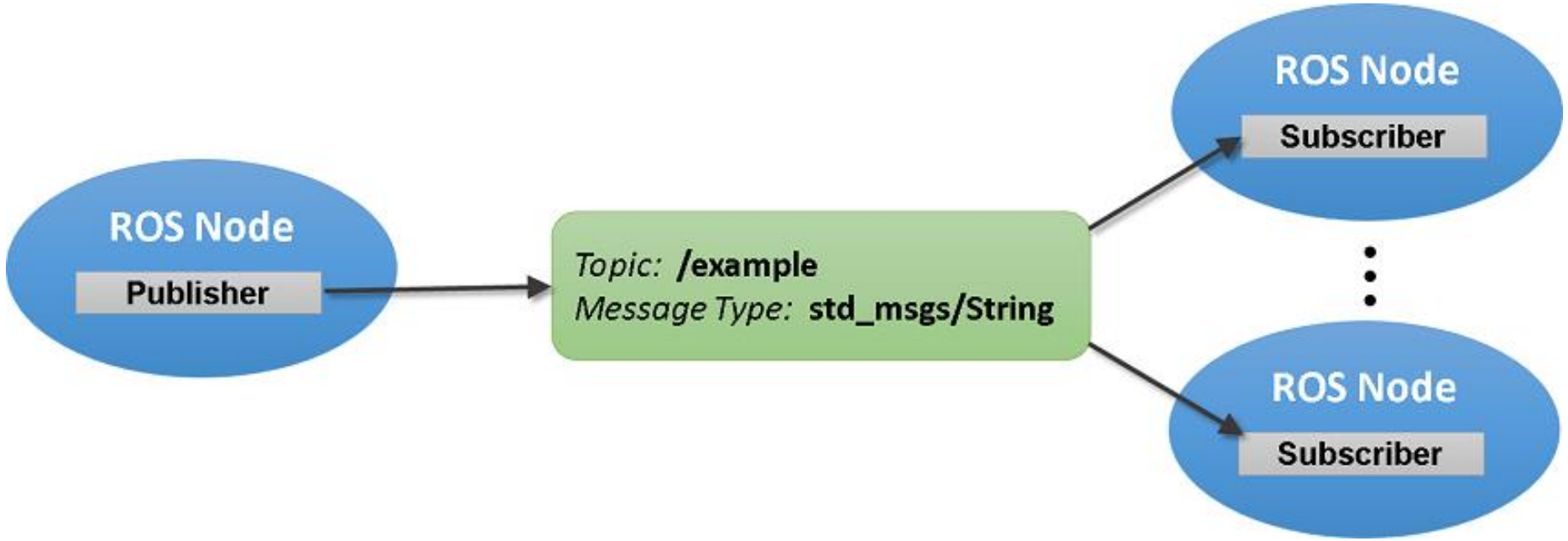
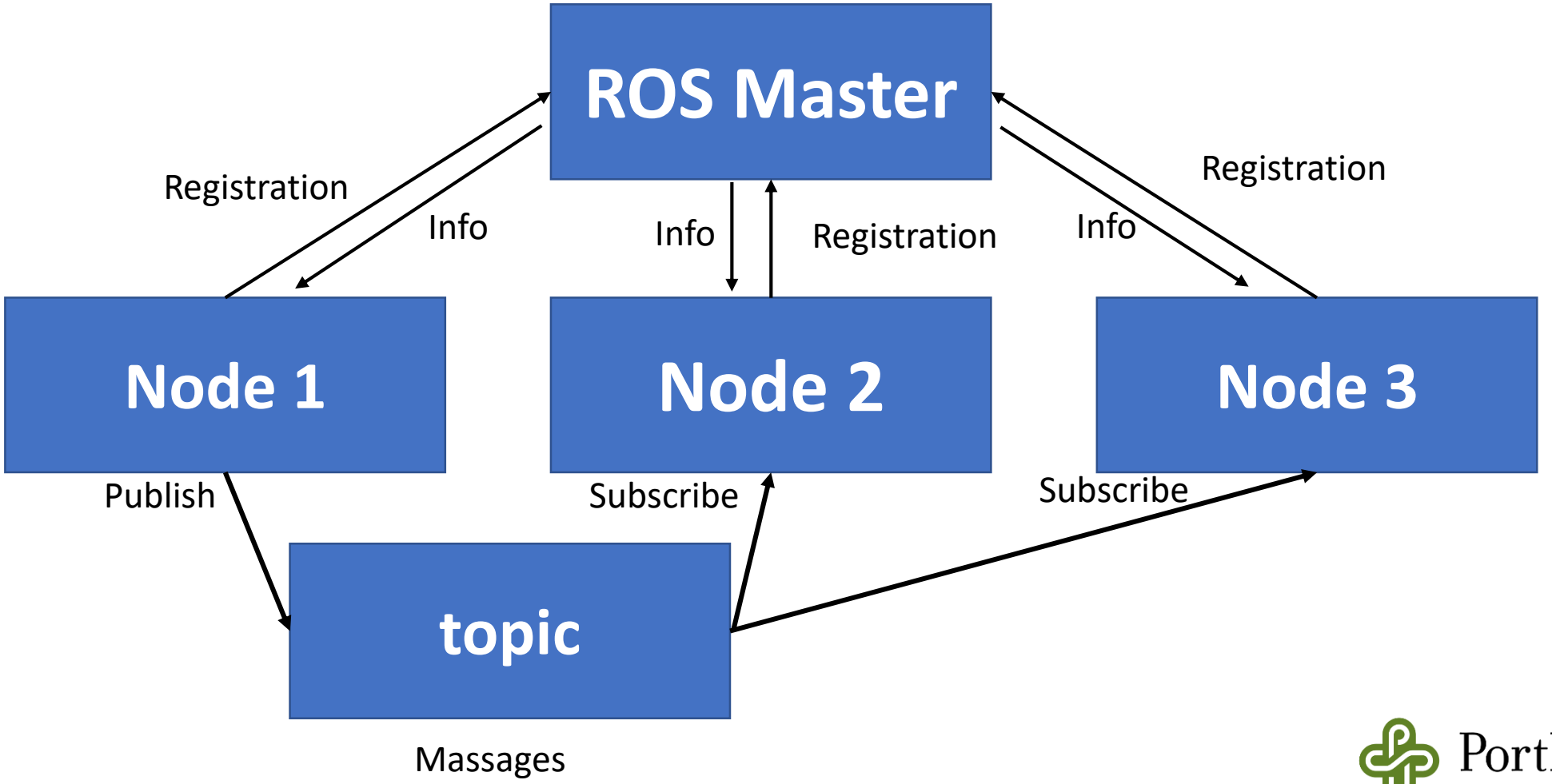


Image Source Link

Portland State UNIVERSITY

ROS Master and Nodes



Portland State UNIVERSITY

ROS File System

Portland State
UNIVERSITY

XML Format Review

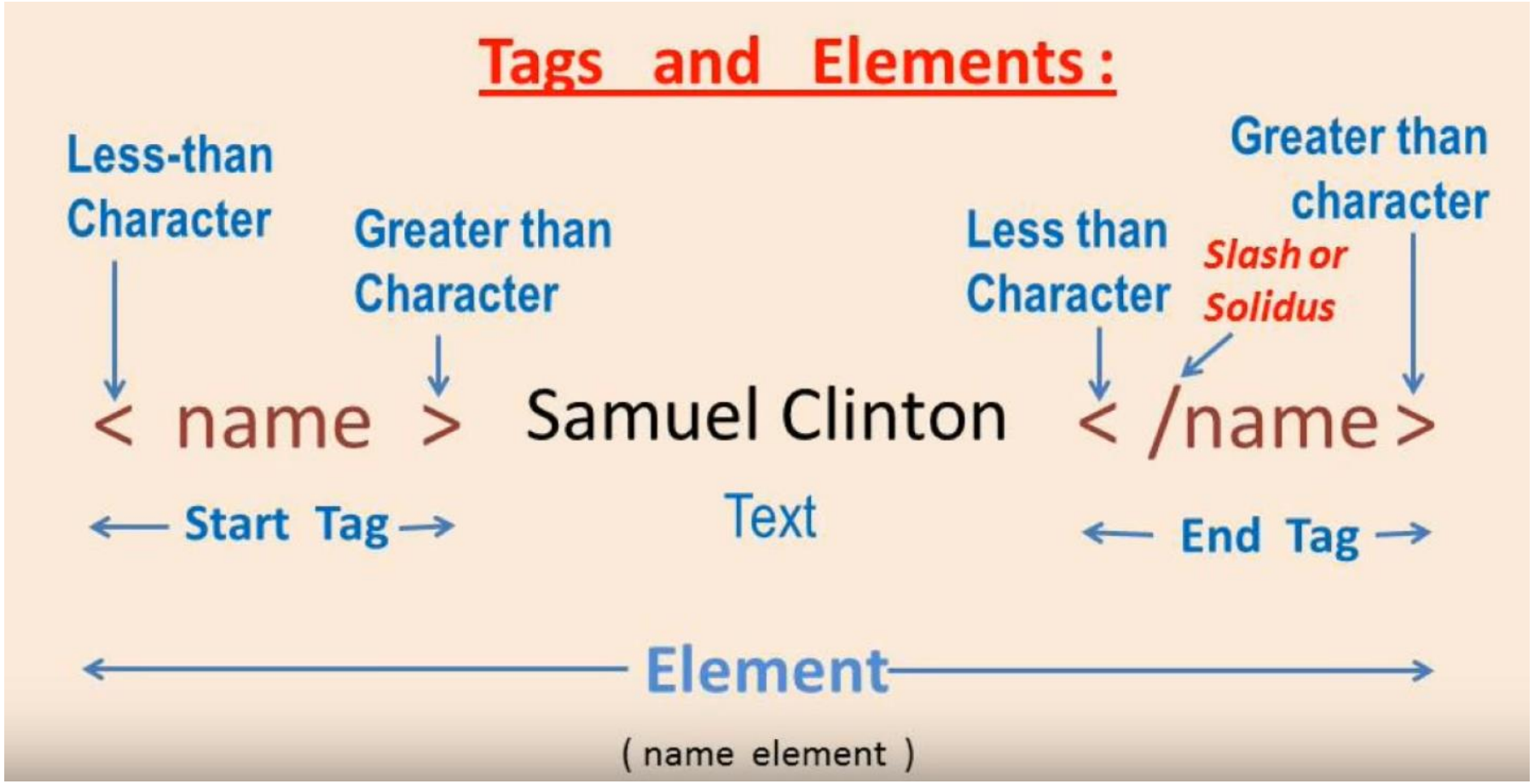
- XML stands for extensible markup language.
- XML documents must contain one **root** element that is the **parent** of all other elements

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

The XML above does not DO anything. XML is just information wrapped in tags.

Portland State UNIVERSITY

XML Tags and Elements



https://www.youtube.com/watch?v=nyk8QO08grM

Portland State UNIVERSITY

Why XML?

- It simplifies data sharing
- It simplifies data transport
- It simplifies platform changes
- It simplifies data availability

If you want to learn more about xml:

https://www.w3schools.com/xml/default.asp

ROS File System

- **Packages:** Packages are the software organization unit of ROS code. Each package can contain libraries, executables, scripts, or other artifacts.
- **Manifests (package.xml):** A manifest is a description of a *package*. It serves to define dependencies between *packages* and to capture meta information about the *package* like version, maintainer, license, etc...

rospack

- **rospack** : gives you information about the package

- **rospack find package_name**
- ex: rospack find roscpp

roscd

- It is used to change directory to a package or a stack.
  - **roscd** **package_name**
- You can also use roscd to move into sub directories.
  - **roscd** **package_name/sub_directory**

rosls

- It is used to list to directories inside a package.
  - **rosls roscpp_tutorials**
    - cmake launch package.xml srv

rosdep

- It is used to install system dependencies.
- **rosdep *install*** my_package
- **rosdep *install*** --from-paths src --ignore-src --rosdistro indigo

Rospack Dependencies

- catkin_create_pkg beginner_tutorials std_msgs rospy roscpp

```
$ rospack depends1 beginner_tutorials
```

```
std_msgs
rospy
roscpp
```

Rospack Dependencies

These dependencies for a package are stored in the **package.xml** file:

```
$ roscd beginner_tutorials
$ cat package.xml
```

```
<package>
...
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
...
</package>
```

Indirect Dependencies

- A dependency can also have its own dependencies.

```
$ rospack depends1 rospy
```

```
genpy
rosgraph
rosgraph_msgs
roslib
std_msgs
```

```
$ rospack depends beginner_tutorials
cpp_common
rostime
roscpp_traits
roscpp_serialization
genmsg
genpy
message_runtime
rosconsole
std_msgs
rosgraph_msgs
xmlrpcpp
roscpp
rosgraph
catkin
rospack
roslib
rospy
```

- All indirect dependencies

Install Dependencies of a Package

- Use rosdep and catkin to build any package in the ROS repository.
- Say our package is named TEST_PACKAGE

- **rosdep install TEST_PACKAGE**

How to Install ROS Packages?

- There are two ways to do it.

- If there is a Debian package it is easy to install. (Method 1)
- If there is no Debian package, you have to install it from sources. (Method 2)

ROS Package Review

- Software in ROS is organized in packages.
- A package can contain your ROS nodes, any libraries, a dataset, configuration files, message types, scripts, service types, some third-party software.
- When you want to create a project in ROS, first you need to create a package
- ROS package are usually for certain ROS distributions
- Packages have dependencies
- http://www.ros.org/browse/list.php?package_type=package&distro=kinetic

Method 1

- **apt-get install ros-<ros-distribution>-<package-name>**

- Example:

- http://www.ros.org/browse/list.php
- **apt-get install ros-indigo-effort-controllers**

Portland State UNIVERSITY

Method 2

- **cd catkin_ws/src**

- https://github.com/ros/ros_tutorials

- git clone -b indigo-devel https://github.com/ros/ros_tutorials.git

- cd catkin_ws

- rosdep install --from-paths src --ignore-src --rosdistro kinetic

- catkin_make

Portland State
UNIVERSITY

Creating a Package

How to Create a ROS Package?

- For a package to be considered a catkin package it must meet a few requirements:
- The package must contain a catkin compliant **package.xml** file.
  - That package.xml file provides meta information about the package.
- The package must contain a **CMakeLists.txt** which uses catkin. If it is a catkin metapackage it must have the relevant boilerplate CMakeLists.txt file.
- There can be no more than one package in each folder.
  - This means no nested packages nor multiple packages sharing the same directory.

ROS Package – Package.xml

Toggle line numbers

```
 1 <?xml version="1.0"?>
 2 <package>
 3   <name>beginner_tutorials</name>
 4   <version>0.1.0</version>
 5   <description>The beginner_tutorials package</description>
 6
 7   <maintainer email="you@yourdomain.tld">Your Name</maintainer>
 8   <license>BSD</license>
 9   <url type="website">http://wiki.ros.org/beginner_tutorials</url>
10   <author email="you@yourdomain.tld">Jane Doe</author>
11
12   <buildtool_depend>catkin</buildtool_depend>
13
14   <build_depend>roscpp</build_depend>
15   <build_depend>rospy</build_depend>
16   <build_depend>std_msgs</build_depend>
17
18   <run_depend>roscpp</run_depend>
19   <run_depend>rospy</run_depend>
20   <run_depend>std_msgs</run_depend>
21
22 </package>
```

Portland State UNIVERSITY

ROS Package

Way 1:

```
my_package/
  CMakeLists.txt
  package.xml
```

Way 2:

```
workspace_folder/          -- WORKSPACE
  src/                     -- SOURCE SPACE
    CMakeLists.txt         -- 'Toplevel' CMake file, provided by catkin
    package_1/
      CMakeLists.txt       -- CMakeLists.txt file for package_1
      package.xml          -- Package manifest for package_1
    ...
    package_n/
      CMakeLists.txt       -- CMakeLists.txt file for package_n
      package.xml          -- Package manifest for package_n
```

Portland State UNIVERSITY

How to Create a ROS Package?

- Create your workspace
- Go to your source directory
  - **cd ~/catkin_ws/src**
- Create a new package
  - **catkin_create_pkg <package_name> [depend1] [depend2] [depend3]**
- Build the workspace and add the workspace to your ROS environment by sourcing the setup.bash file
  - **cd ~/catkin_ws**
  - **. ~/catkin_ws/devel/setup.bash**

Portland State UNIVERSITY

Customize your ROS Package

The generated package.xml should be in your new package.

Toggle line numbers

```
 1 <?xml version="1.0"?>
 2 <package>
 3   <name>beginner_tutorials</name>
 4   <version>0.1.0</version>
 5   <description>The beginner_tutorials package</description>
 6
 7   <maintainer email="you@yourdomain.tld">Your Name</maintainer>
 8   <license>BSD</license>
 9   <url type="website">http://wiki.ros.org/beginner_tutorials</url>
10   <author email="you@yourdomain.tld">Jane Doe</author>
11
12   <buildtool_depend>catkin</buildtool_depend>
13
14   <build_depend>roscpp</build_depend>
15   <build_depend>rospy</build_depend>
16   <build_depend>std_msgs</build_depend>
17
18   <run_depend>roscpp</run_depend>
19   <run_depend>rospy</run_depend>
20   <run_depend>std_msgs</run_depend>
21
22 </package>
```

Portland State UNIVERSITY

Customize your ROS Package

- **Description Tag:**
  - Describe your package in a few words

Toggle line numbers

```
5    <description>The beginner_tutorials package</description>
```

Customize your ROS Package

- **Maintainer Tags:**
  - This is a required and important tag for the package.xml
  - Let others know who to contact about the package.

Toggle line numbers

```
7   <!-- One maintainer tag required, multiple allowed, one person per tag -->
8   <!-- Example:  -->
9   <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
10  <maintainer email="user@todo.todo">user</maintainer>
```

Portland State UNIVERSITY

Customize your ROS Package

- **License Tags:**
  - You should choose a license and fill it in here. Some common open source licenses are BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, and LGPLv3

```
Toggle line numbers

    8    <license>BSD</license>
```

Portland State UNIVERSITY

Customize your ROS Package

- **Dependencies Tags:**
  - We define all of our specified dependencies to be available at build and run time.

Toggle line numbers

```
12   <buildtool_depend>catkin</buildtool_depend>
13
14   <build_depend>roscpp</build_depend>
15   <build_depend>rospy</build_depend>
16   <build_depend>std_msgs</build_depend>
17
18   <run_depend>roscpp</run_depend>
19   <run_depend>rospy</run_depend>
20   <run_depend>std_msgs</run_depend>
```

Portland State UNIVERSITY

ROS Package Review

Toggle line numbers

```
1  <?xml version="1.0"?>
2  <package>
3    <name>beginner_tutorials</name>
4    <version>0.1.0</version>
5    <description>The beginner_tutorials package</description>
6
7    <maintainer email="you@yourdomain.tld">Your Name</maintainer>
8    <license>BSD</license>
9    <url type="website">http://wiki.ros.org/beginner_tutorials</url>
10   <author email="you@yourdomain.tld">Jane Doe</author>
11
12   <buildtool_depend>catkin</buildtool_depend>
13
14   <build_depend>roscpp</build_depend>
15   <build_depend>rospy</build_depend>
16   <build_depend>std_msgs</build_depend>
17
18   <run_depend>roscpp</run_depend>
19   <run_depend>rospy</run_depend>
20   <run_depend>std_msgs</run_depend>
21
22 </package>
```

1

2

3

4

5

Portland State UNIVERSITY

Creating a Subscriber Node

Portland State
UNIVERSITY

Creating a Subscriber Node

- Go to your package in workspace
  - **cd catkin_ws/src/mypackage**
- Create directory called “scripts”
  - **mkdir scripts**
- Go in to scripts
  - **cd scripts**
- Create script with your favirote editor “listener.py”
- Make the python script executable
  - **chmod +x listener.py**

Portland State UNIVERSITY

Creating a Subscriber Node

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    rospy.spin()

if __name__ == '__main__':
    listener()
```

Portland State UNIVERSITY

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)

def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # node are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('listener', anonymous=True)

    rospy.Subscriber("chatter", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

It tells your system this a Python file
It allows us to use Python with ROS
Import message type string, so we can reuse it

It prints to screen as it also writes to stdout

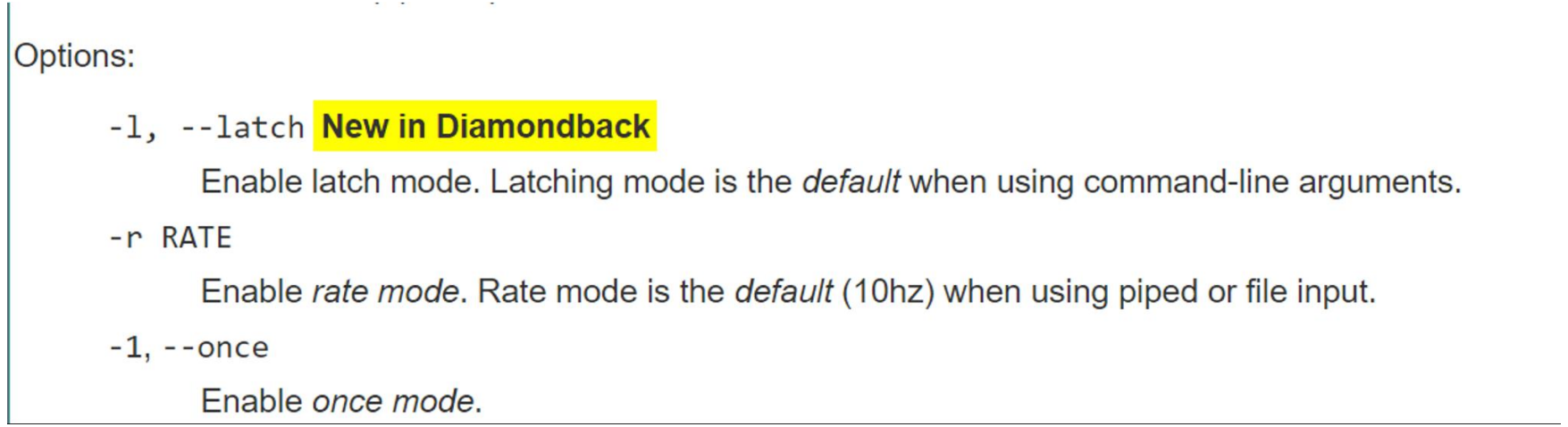
Create a node called listener

Node subscribes to the chatter.
When new messages are received
callback method is invoked.

1 2 3

Let’s Test Our Node

- Publishes data to a topic
- rostopic pub /topic_name std_msgs/String Perkowski



Options:

`-l, --latch` **New in Diamondback**

Enable latch mode. Latching mode is the *default* when using command-line arguments.

`-r RATE`

Enable *rate mode*. Rate mode is the *default* (10hz) when using piped or file input.

`-1, --once`

Enable *once mode*.

- Publishes data to a topic
- rostopic pub /TA_topic std_msgs/String Hello World

Portland State UNIVERSITY