**IE 410 – INTRODUCTION TO ROBOTICS**

**Lab-5 report**

**Running talker and listener codes**

**Team M410:**

201901011 – HIMANSHU DUDHATRA

201901024 – DHAVALSINH RAJ

201901100 – SHUBHAM PATEL

201901145 – GARGEY PATEL

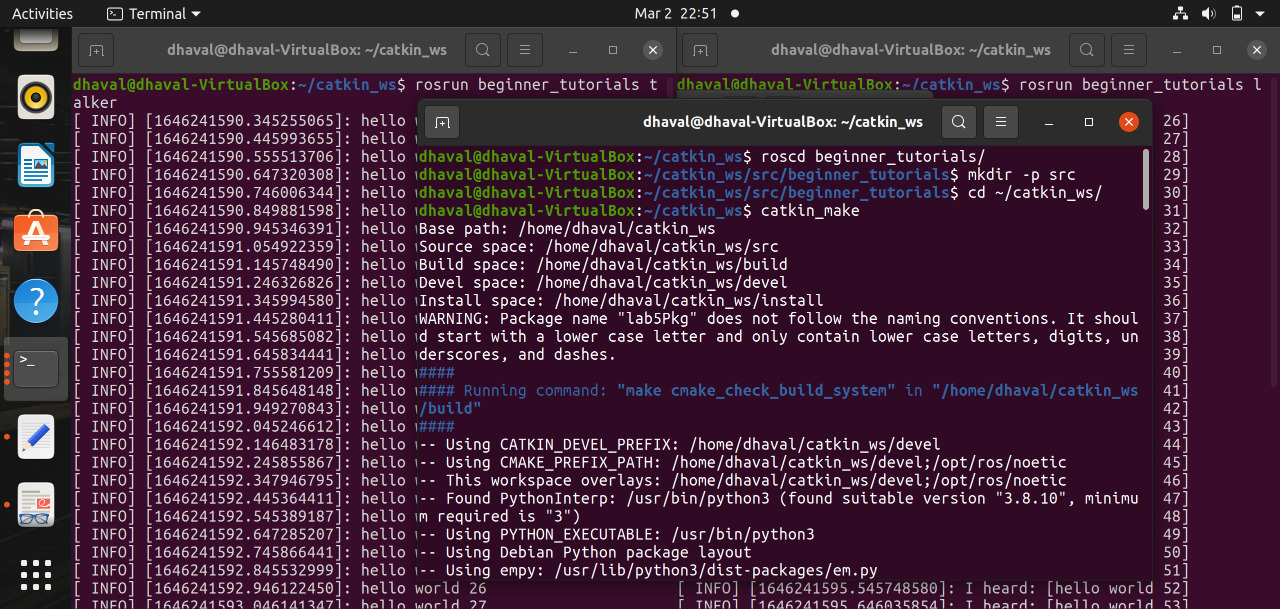
**CPP**

* **Change directories to your beginner\_tutorials package you created in your catkin workspace previous tutorials:**

roscd beginner\_tutorials

* **Create a**src**directory in the beginner\_tutorials package directory:**

mkdir -p src



* **Talker.cpp**

/\*

\* Copyright (C) 2008, Morgan Quigley and Willow Garage, Inc.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions are met:

\* \* Redistributions of source code must retain the above copyright notice,

\* this list of conditions and the following disclaimer.

\* \* Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\* \* Neither the names of Stanford University or Willow Garage, Inc. nor the names of its

\* contributors may be used to endorse or promote products derived from

\* this software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

\* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

\* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE

\* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

\* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

\* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

\* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

\* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

\* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

\* POSSIBILITY OF SUCH DAMAGE.

\*/

// %Tag(FULLTEXT)%

// %Tag(ROS\_HEADER)%

#include "ros/ros.h"

// %EndTag(ROS\_HEADER)%

// %Tag(MSG\_HEADER)%

#include "std\_msgs/String.h"

// %EndTag(MSG\_HEADER)%

#include <sstream>

/\*\*

\* This tutorial demonstrates simple sending of messages over the ROS system.

\*/

int main(int argc, char \*\*argv)

{

/\*\*

\* The ros::init() function needs to see argc and argv so that it can perform

\* any ROS arguments and name remapping that were provided at the command line.

\* For programmatic remappings you can use a different version of init() which takes

\* remappings directly, but for most command-line programs, passing argc and argv is

\* the easiest way to do it. The third argument to init() is the name of the node.

\*

\* You must call one of the versions of ros::init() before using any other

\* part of the ROS system.

\*/

// %Tag(INIT)%

ros::init(argc, argv, "talker");

// %EndTag(INIT)%

/\*\*

\* NodeHandle is the main access point to communications with the ROS system.

\* The first NodeHandle constructed will fully initialize this node, and the last

\* NodeHandle destructed will close down the node.

\*/

// %Tag(NODEHANDLE)%

ros::NodeHandle n;

// %EndTag(NODEHANDLE)%

/\*\*

\* The advertise() function is how you tell ROS that you want to

\* publish on a given topic name. This invokes a call to the ROS

\* master node, which keeps a registry of who is publishing and who

\* is subscribing. After this advertise() call is made, the master

\* node will notify anyone who is trying to subscribe to this topic name,

\* and they will in turn negotiate a peer-to-peer connection with this

\* node. advertise() returns a Publisher object which allows you to

\* publish messages on that topic through a call to publish(). Once

\* all copies of the returned Publisher object are destroyed, the topic

\* will be automatically unadvertised.

\*

\* The second parameter to advertise() is the size of the message queue

\* used for publishing messages. If messages are published more quickly

\* than we can send them, the number here specifies how many messages to

\* buffer up before throwing some away.

\*/

// %Tag(PUBLISHER)%

ros::Publisher chatter\_pub = n.advertise<std\_msgs::String>("chatter", 1000);

// %EndTag(PUBLISHER)%

// %Tag(LOOP\_RATE)%

ros::Rate loop\_rate(10);

// %EndTag(LOOP\_RATE)%

/\*\*

\* A count of how many messages we have sent. This is used to create

\* a unique string for each message.

\*/

// %Tag(ROS\_OK)%

int count = 0;

while (ros::ok())

{

// %EndTag(ROS\_OK)%

/\*\*

\* This is a message object. You stuff it with data, and then publish it.

\*/

// %Tag(FILL\_MESSAGE)%

std\_msgs::String msg;

std::stringstream ss;

ss << "hello world " << count;

msg.data = ss.str();

// %EndTag(FILL\_MESSAGE)%

// %Tag(ROSCONSOLE)%

ROS\_INFO("%s", msg.data.c\_str());

// %EndTag(ROSCONSOLE)%

/\*\*

\* The publish() function is how you send messages. The parameter

\* is the message object. The type of this object must agree with the type

\* given as a template parameter to the advertise<>() call, as was done

\* in the constructor above.

\*/

// %Tag(PUBLISH)%

chatter\_pub.publish(msg);

// %EndTag(PUBLISH)%

// %Tag(SPINONCE)%

ros::spinOnce();

// %EndTag(SPINONCE)%

// %Tag(RATE\_SLEEP)%

loop\_rate.sleep();

// %EndTag(RATE\_SLEEP)%

++count;

}

return 0;

}

// %EndTag(FULLTEXT)%

* **Listner.cpp**

/\*

\* Copyright (C) 2008, Morgan Quigley and Willow Garage, Inc.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions are met:

\* \* Redistributions of source code must retain the above copyright notice,

\* this list of conditions and the following disclaimer.

\* \* Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\* \* Neither the names of Stanford University or Willow Garage, Inc. nor the names of its

\* contributors may be used to endorse or promote products derived from

\* this software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

\* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

\* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE

\* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

\* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

\* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

\* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN

\* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)

\* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

\* POSSIBILITY OF SUCH DAMAGE.

\*/

// %Tag(FULLTEXT)%

#include "ros/ros.h"

#include "std\_msgs/String.h"

/\*\*

\* This tutorial demonstrates simple receipt of messages over the ROS system.

\*/

// %Tag(CALLBACK)%

void chatterCallback(const std\_msgs::String::ConstPtr& msg)

{

ROS\_INFO("I heard: [%s]", msg->data.c\_str());

}

// %EndTag(CALLBACK)%

int main(int argc, char \*\*argv)

{

/\*\*

\* The ros::init() function needs to see argc and argv so that it can perform

\* any ROS arguments and name remapping that were provided at the command line.

\* For programmatic remappings you can use a different version of init() which takes

\* remappings directly, but for most command-line programs, passing argc and argv is

\* the easiest way to do it. The third argument to init() is the name of the node.

\*

\* You must call one of the versions of ros::init() before using any other

\* part of the ROS system.

\*/

ros::init(argc, argv, "listener");

/\*\*

\* NodeHandle is the main access point to communications with the ROS system.

\* The first NodeHandle constructed will fully initialize this node, and the last

\* NodeHandle destructed will close down the node.

\*/

ros::NodeHandle n;

/\*\*

\* The subscribe() call is how you tell ROS that you want to receive messages

\* on a given topic. This invokes a call to the ROS

\* master node, which keeps a registry of who is publishing and who

\* is subscribing. Messages are passed to a callback function, here

\* called chatterCallback. subscribe() returns a Subscriber object that you

\* must hold on to until you want to unsubscribe. When all copies of the Subscriber

\* object go out of scope, this callback will automatically be unsubscribed from

\* this topic.

\*

\* The second parameter to the subscribe() function is the size of the message

\* queue. If messages are arriving faster than they are being processed, this

\* is the number of messages that will be buffered up before beginning to throw

\* away the oldest ones.

\*/

// %Tag(SUBSCRIBER)%

ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);

// %EndTag(SUBSCRIBER)%

/\*\*

\* ros::spin() will enter a loop, pumping callbacks. With this version, all

\* callbacks will be called from within this thread (the main one). ros::spin()

\* will exit when Ctrl-C is pressed, or the node is shutdown by the master.

\*/

// %Tag(SPIN)%

ros::spin();

// %EndTag(SPIN)%

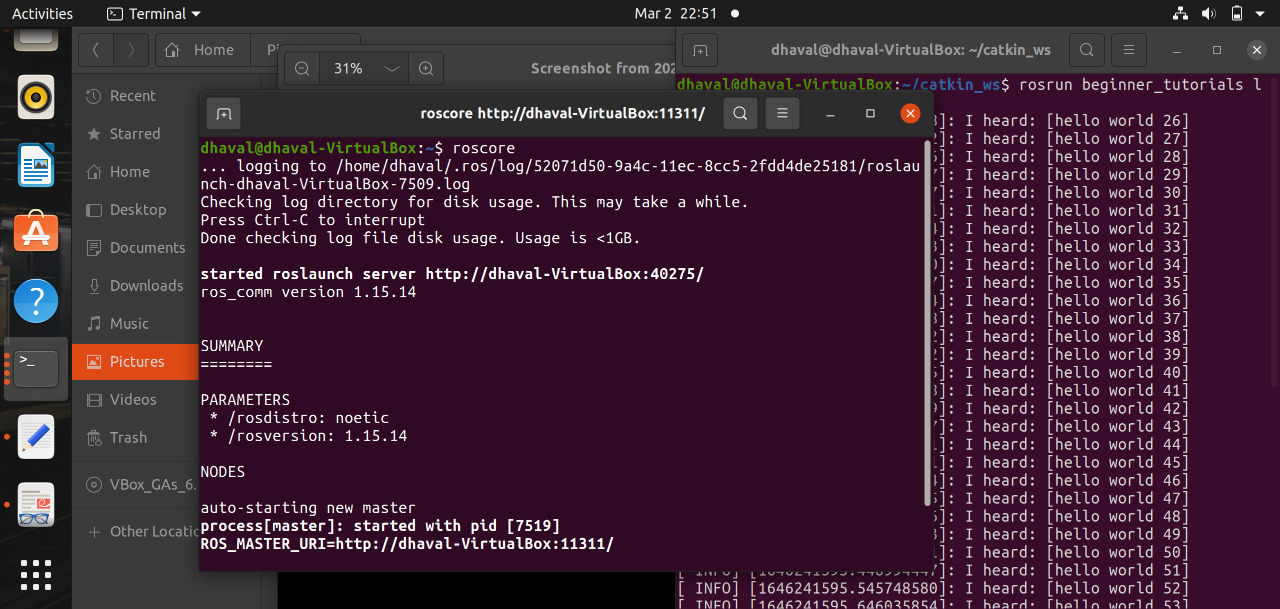
return 0;

}

// %EndTag(FULLTEXT)%

* **Now we will run** $ roscore **command in new terminal**

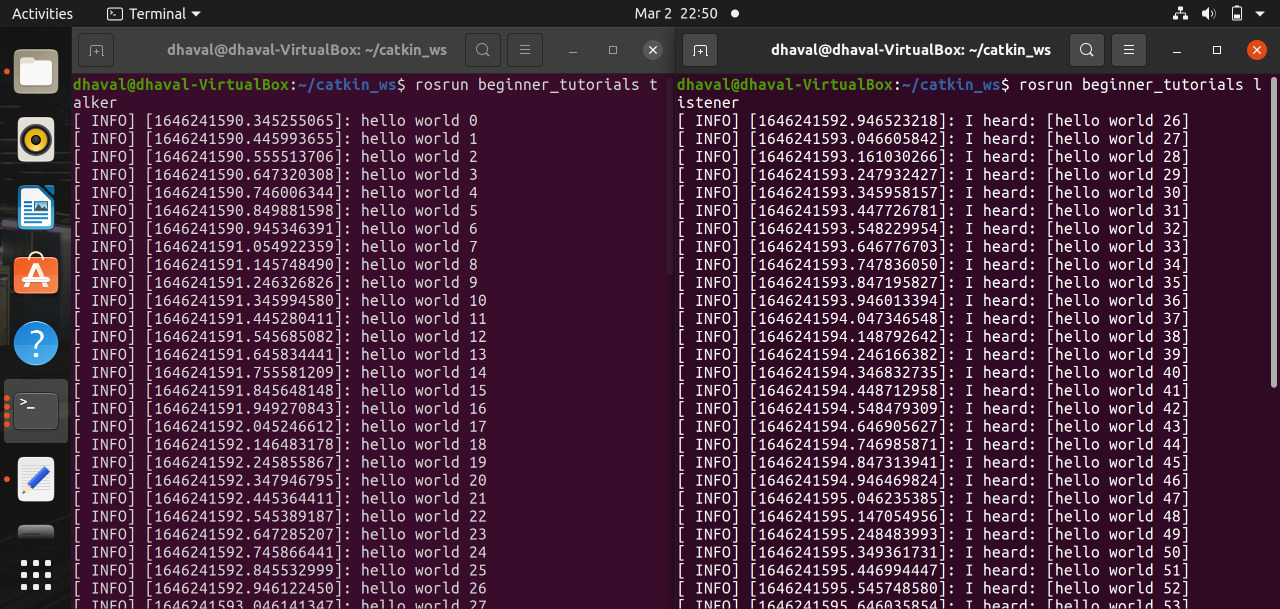
$ roscore



* **Now we will run publisher and subscriber in different terminal windows**

$ rosrun beginner\_tutorials talker

$ rosrun beginner\_tutorials listener

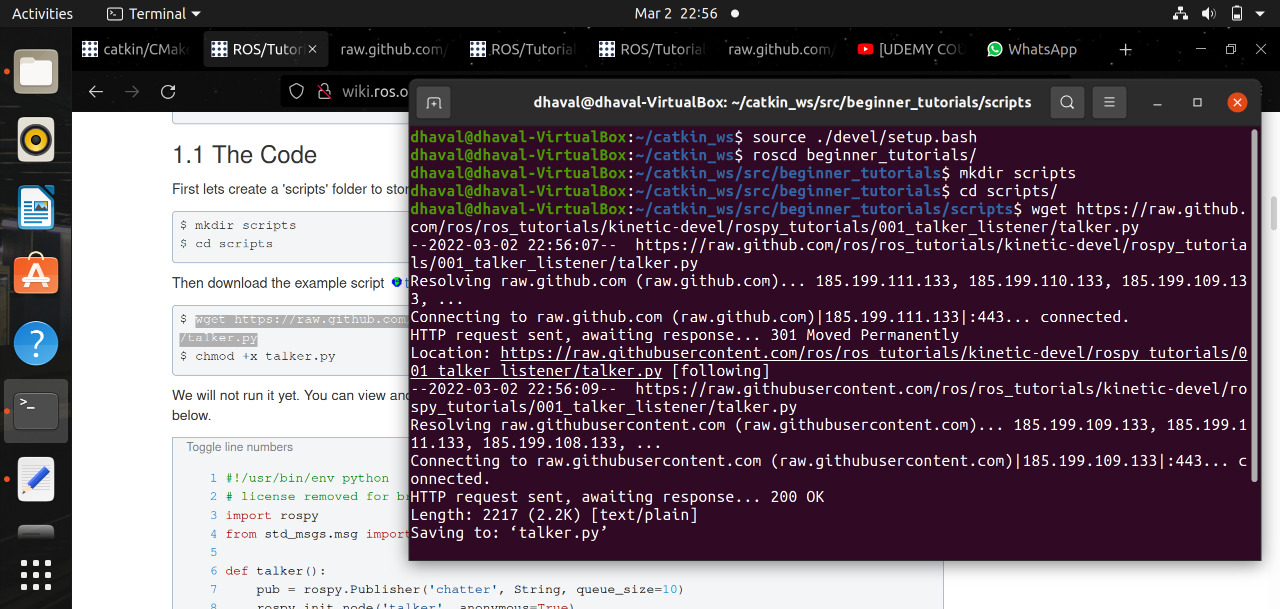


**PYTHON**

* **First lets create a 'scripts' folder to store our Python scripts in:**

$ mkdir scripts

$ cd scripts

****

* **Talker.py**

#!/usr/bin/env python

# Software License Agreement (BSD License)

#

# Copyright (c) 2008, Willow Garage, Inc.

# All rights reserved.

#

# Redistribution and use in source and binary forms, with or without

# modification, are permitted provided that the following conditions

# are met:

#

# \* Redistributions of source code must retain the above copyright

# notice, this list of conditions and the following disclaimer.

# \* Redistributions in binary form must reproduce the above

# copyright notice, this list of conditions and the following

# disclaimer in the documentation and/or other materials provided

# with the distribution.

# \* Neither the name of Willow Garage, Inc. nor the names of its

# contributors may be used to endorse or promote products derived

# from this software without specific prior written permission.

#

# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS

# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE

# COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,

# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,

# BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT

# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN

# ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

# POSSIBILITY OF SUCH DAMAGE.

#

# Revision $Id$

## Simple talker demo that published std\_msgs/Strings messages

## to the 'chatter' topic

import rospy

from std\_msgs.msg import String

def talker():

pub = rospy.Publisher('chatter', String, queue\_size=10)

rospy.init\_node('talker', anonymous=True)

rate = rospy.Rate(10) # 10hz

while not rospy.is\_shutdown():

hello\_str = "hello world %s" % rospy.get\_time()

rospy.loginfo(hello\_str)

pub.publish(hello\_str)

rate.sleep()

if \_\_name\_\_ == '\_\_main\_\_':

try:

talker()

except rospy.ROSInterruptException:

pass

* **Lister.py**

#!/usr/bin/env python

# Software License Agreement (BSD License)

#

# Copyright (c) 2008, Willow Garage, Inc.

# All rights reserved.

#

# Redistribution and use in source and binary forms, with or without

# modification, are permitted provided that the following conditions

# are met:

#

# \* Redistributions of source code must retain the above copyright

# notice, this list of conditions and the following disclaimer.

# \* Redistributions in binary form must reproduce the above

# copyright notice, this list of conditions and the following

# disclaimer in the documentation and/or other materials provided

# with the distribution.

# \* Neither the name of Willow Garage, Inc. nor the names of its

# contributors may be used to endorse or promote products derived

# from this software without specific prior written permission.

#

# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

# LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS

# FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE

# COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,

# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,

# BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

# CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT

# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN

# ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

# POSSIBILITY OF SUCH DAMAGE.

#

# Revision $Id$

## Simple talker demo that listens to std\_msgs/Strings published

## to the 'chatter' topic

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

# name 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()

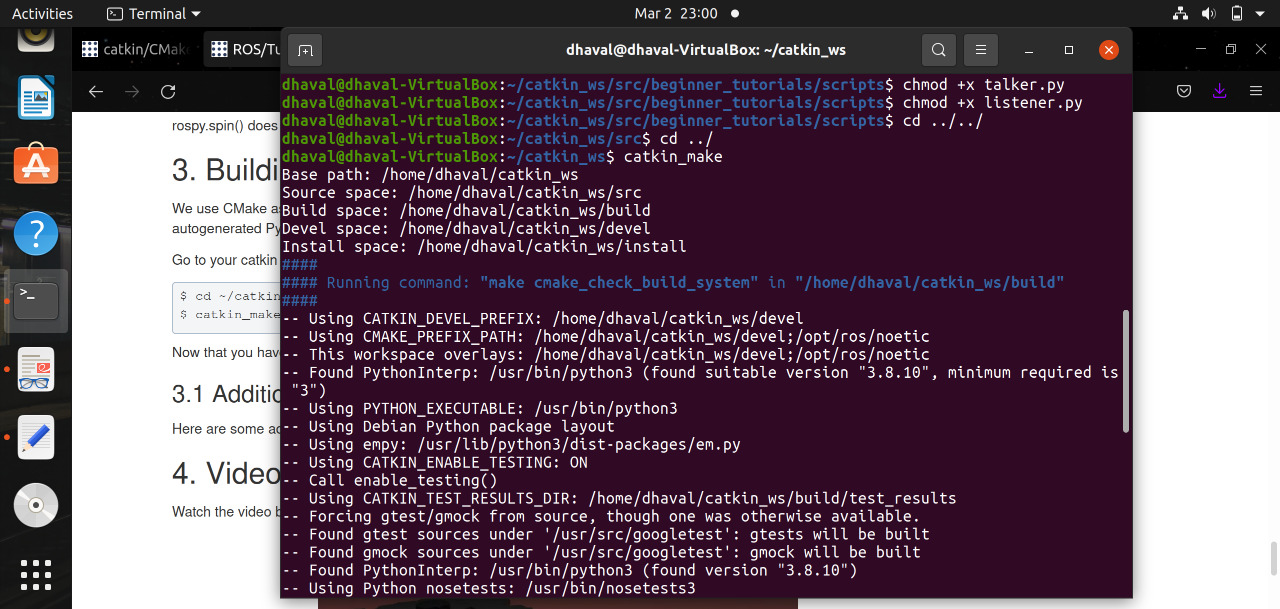
if \_\_name\_\_ == '\_\_main\_\_':

listener()

* **Now making talker.py and listener.py executable:**

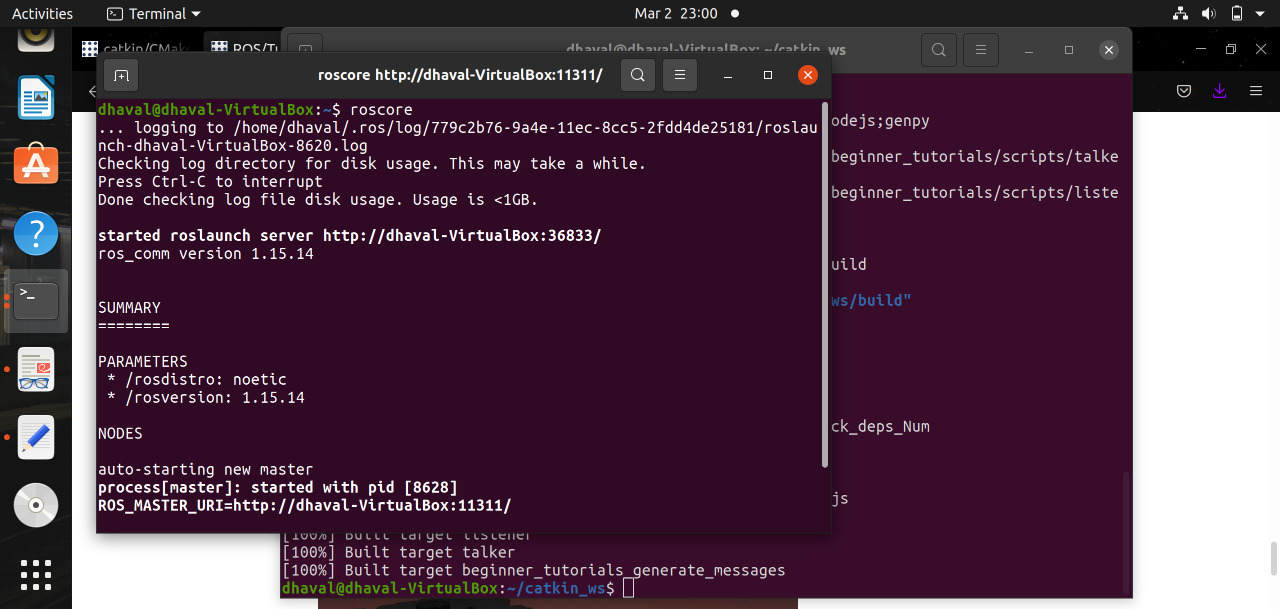
$ chmod +x talker.py

$ chmod +x listener.py

****

* **Now we will run** $ roscore **command in new terminal**

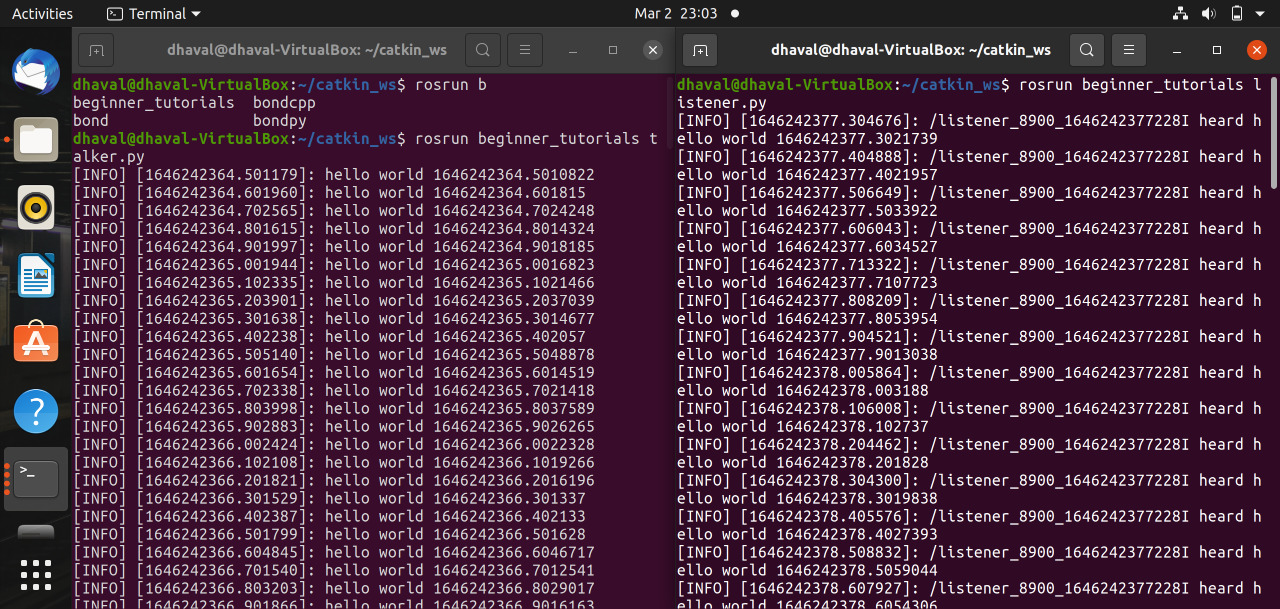
$ roscore

****

* **Now we will run publisher and subscriber in different terminal windows**

$ rosrun beginner\_tutorials talker.py

$ rosrun beginner\_tutorials listener.py

****