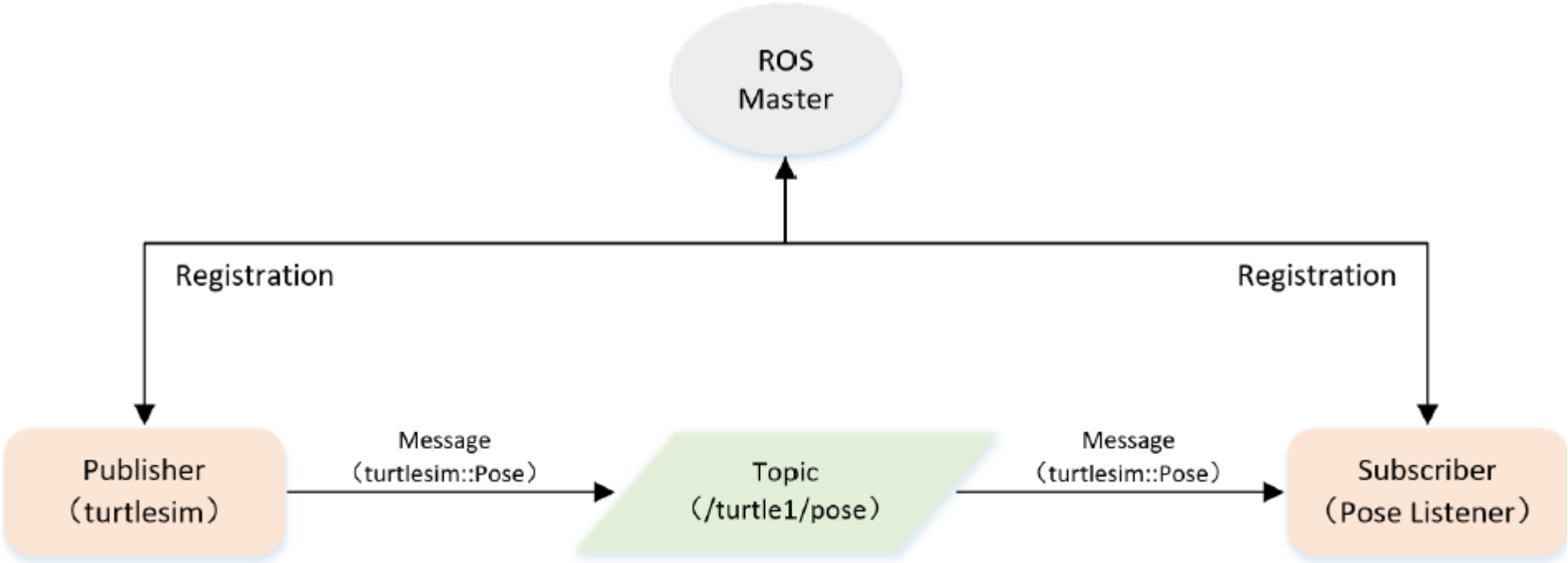
ECE 4703
Mobile Autonomous Robots

Jenny Zhen Yu
zhenyu@cpp.edu
Department of Electrical and Computer Engineering
California State Polytechnic University, Pomona

Lecture 6: Subscriber Programming

Topic


ROS
Master
Registration
Registration
Publisher
（turtlesim）
Message
（turtlesim::Pose）
Topic
（/turtle1/pose）
Message
（turtlesim::Pose）
Subscriber
（Pose Listener）

C++ Code

Open

```
#include <ros/ros.h>
#include "turtlesim/Pose.h"


void poseCallback(const turtlesim::Pose::ConstPtr& msg)
{

    ROS_INFO("Turtle pose: x:%0.6f, y:%0.6f", msg->x, msg->y);
}

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

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


    ros::NodeHandle n;


    ros::Subscriber pose_sub = n.subscribe("/turtle1/pose", 10, poseCallback);


    ros::spin();

    return 0;
}
```

Python Code

Open

pose_subscriber.py

```
1 import rospy
2 from turtlesim.msg import Pose
3
4 def poseCallback(msg):
5     rospy.loginfo("Turtle pose: x:%0.6f, y:%0.6f", msg.x, msg.y)
6
7 def pose_subscriber():
8
9     rospy.init_node('pose_subscriber', anonymous=True)
10
11
12     rospy.Subscriber("/turtle1/pose", Pose, poseCallback)
13
14
15     rospy.spin()
16
17 if __name__ == '__main__':
18     pose_subscriber()
19
20
```

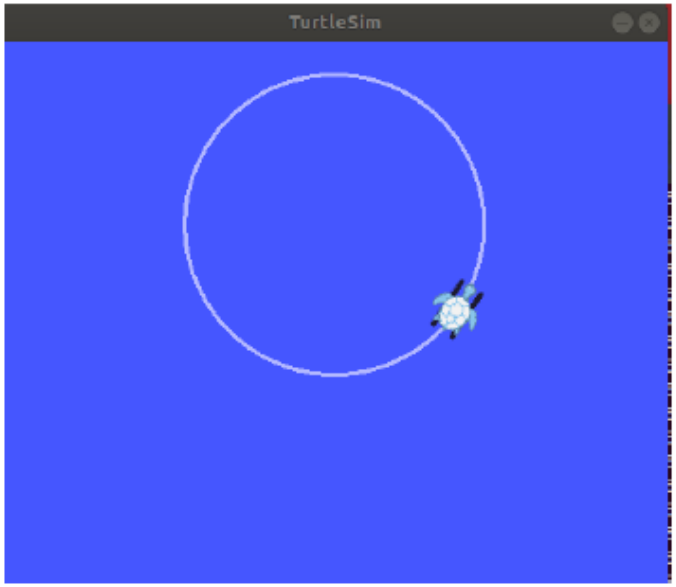
Files

Compiling Code

```
## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/learning_topic_node.cpp)

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

add_executable(velocity_publisher src/velocity_publisher.cpp)
target_link_libraries(velocity_publisher ${catkin_LIBRARIES})


add_executable(pose_subscriber src/pose_subscriber.cpp)
target_link_libraries(pose_subscriber ${catkin_LIBRARIES})
```

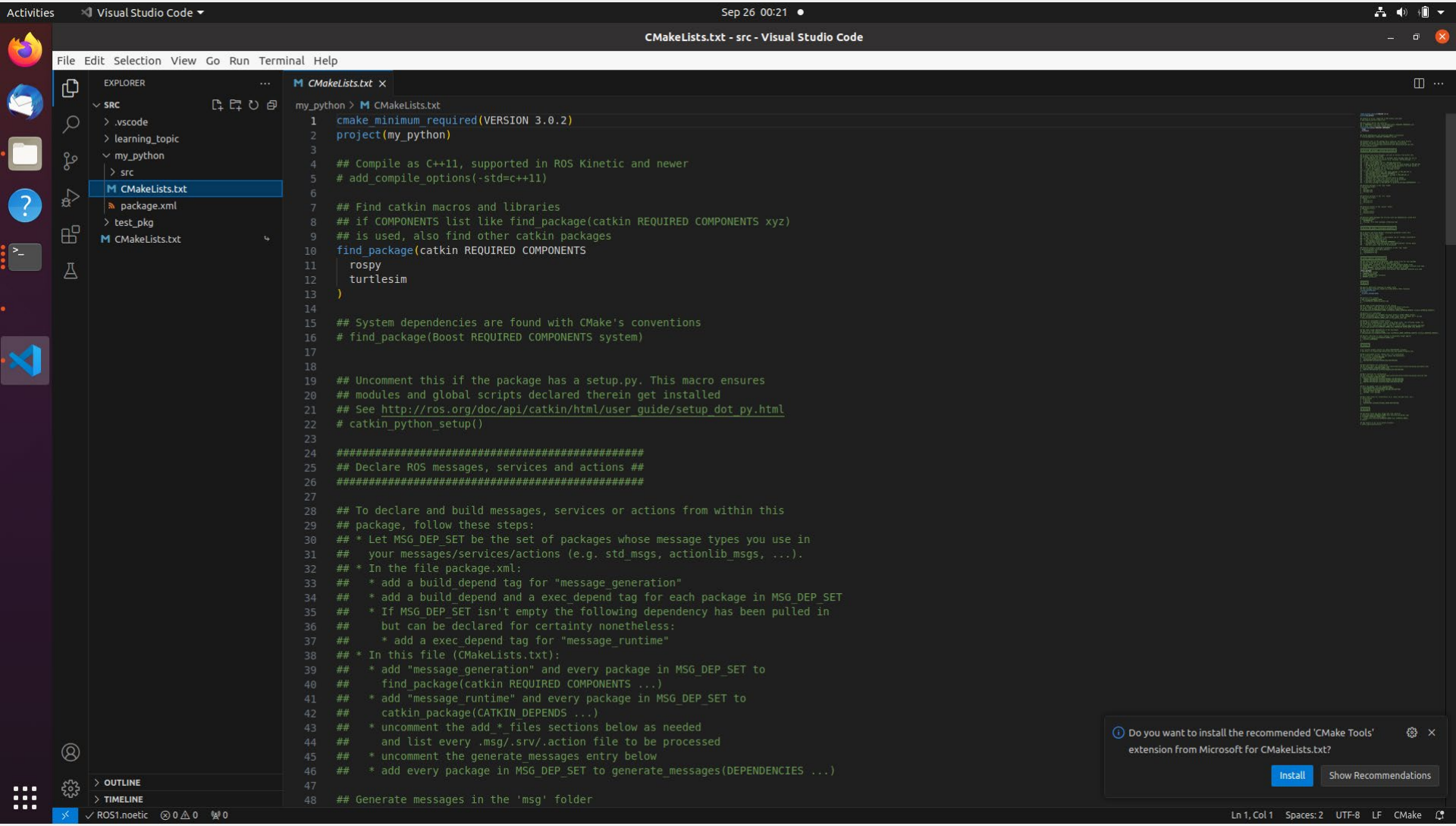
Run Program

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun turtlesim turtlesim_node
$ rosrun learning_topic velocity_publisher
```

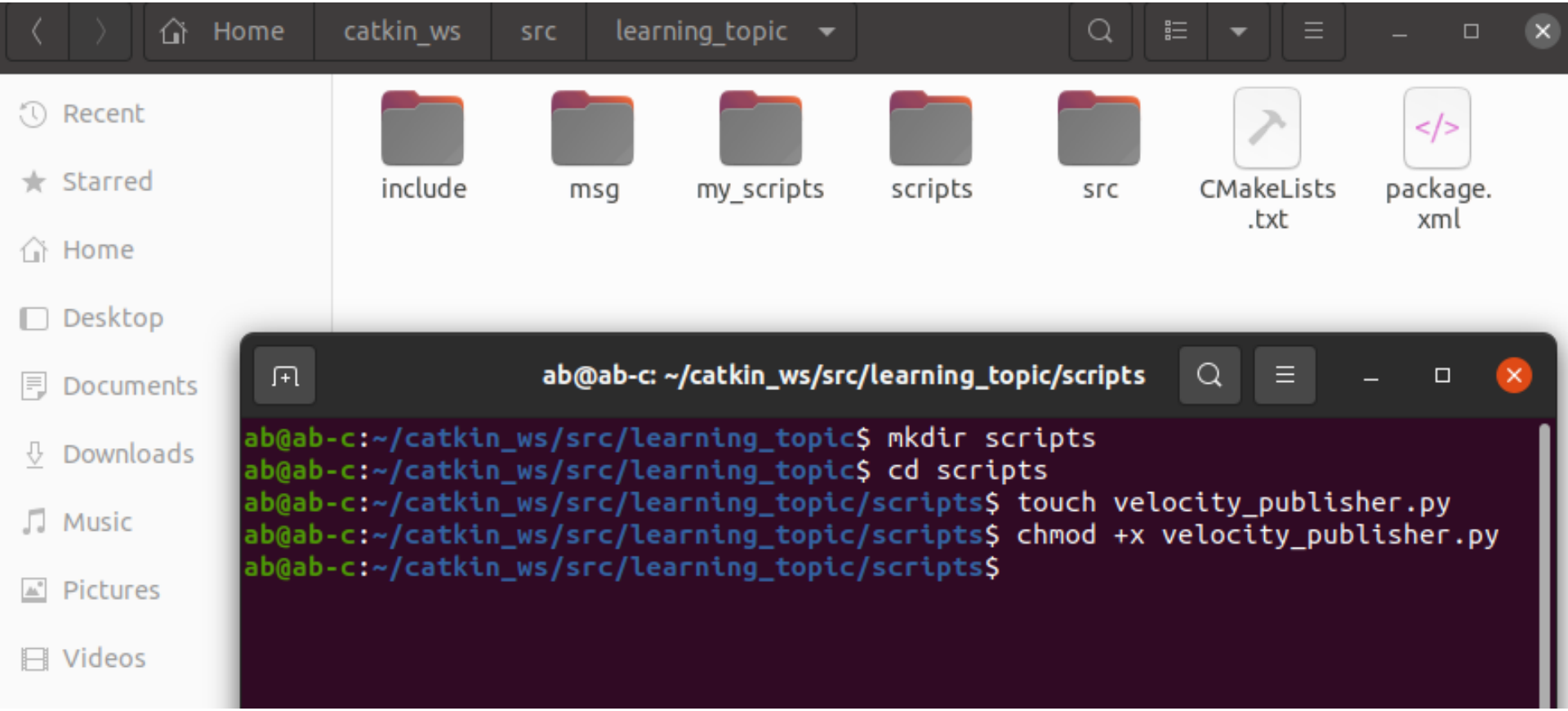


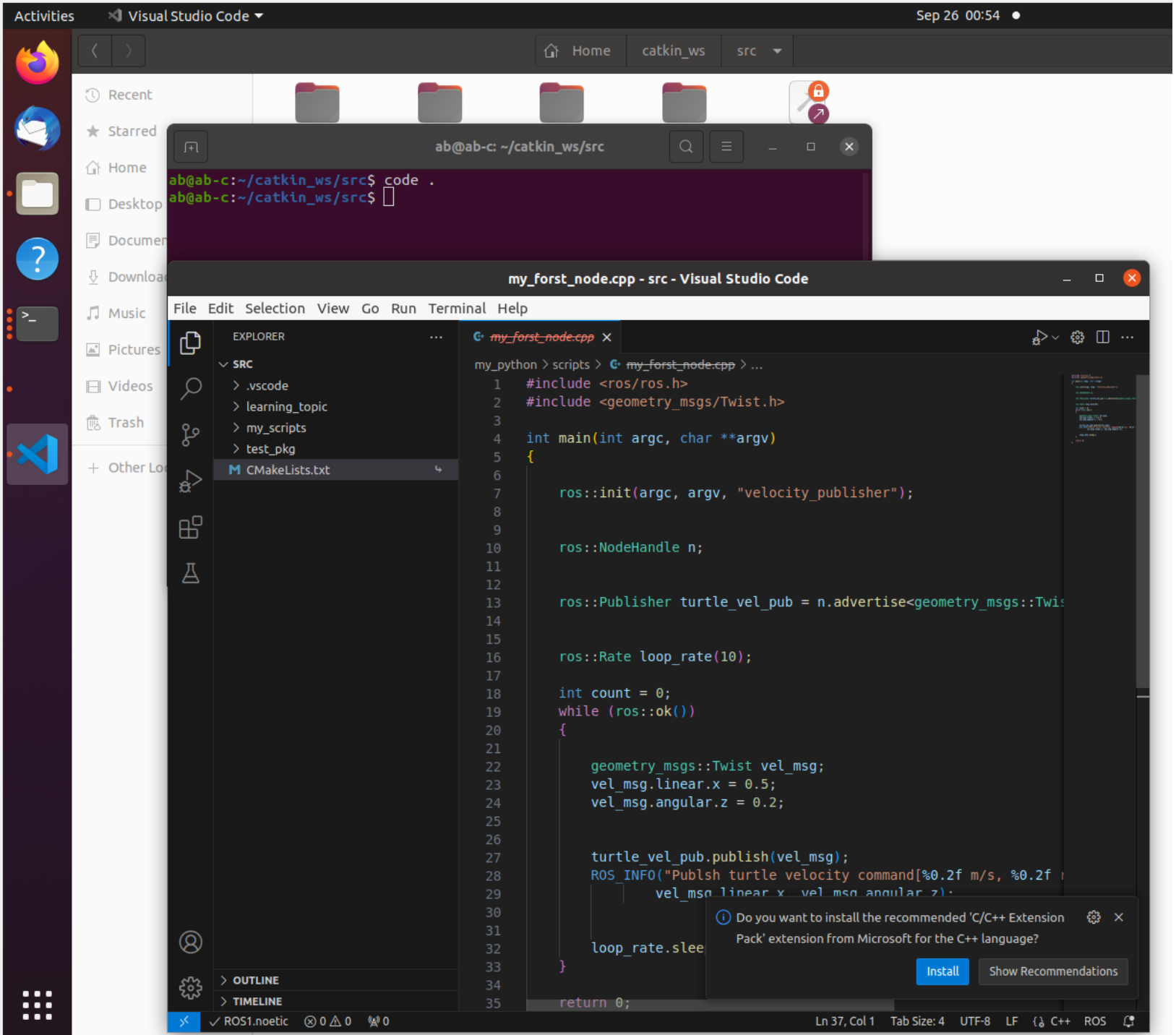

```
hcx@hcx-vpc:~/catkin_ws$ rosrun learning_topic pose_subscriber
[ INFO] [1562211557.322259871]: Turtle pose: x:6.389005, y:10.396028
[ INFO] [1562211557.339097278]: Turtle pose: x:6.381475, y:10.398730
[ INFO] [1562211557.354512018]: Turtle pose: x:6.373938, y:10.401410
[ INFO] [1562211557.370549572]: Turtle pose: x:6.366391, y:10.404065
[ INFO] [1562211557.387085434]: Turtle pose: x:6.358836, y:10.406695
[ INFO] [1562211557.402710847]: Turtle pose: x:6.351273, y:10.409303
[ INFO] [1562211557.418887039]: Turtle pose: x:6.343701, y:10.411885
[ INFO] [1562211557.434469988]: Turtle pose: x:6.336121, y:10.414443
[ INFO] [1562211557.450210135]: Turtle pose: x:6.328533, y:10.416977
[ INFO] [1562211557.465994903]: Turtle pose: x:6.320937, y:10.419487
[ INFO] [1562211557.482173454]: Turtle pose: x:6.313333, y:10.421972
```

Python


Activities
Visual Studio Code
Sep 26 00:21
CMakeLists.txt - src - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
SRC
.vscode
learning_topic
my_python
src
CMakeLists.txt
package.xml
test_pkg
CMakeLists.txt
OUTLINE
TIMELINE
CMakeLists.txt
my_python > CMakeLists.txt
cmake_minimum_required(VERSION 3.0.2)
project(my_python)
Compile as C++11, supported in ROS Kinetic and newer
add_compile_options(-std=c++11)
Find catkin macros and libraries
if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
rospy
turtlesim
)
System dependencies are found with CMake's conventions
find_package(Boost REQUIRED COMPONENTS system)
Uncomment this if the package has a setup.py. This macro ensures
modules and global scripts declared therein get installed
See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
catkin_python_setup()
Declare ROS messages, services and actions
To declare and build messages, services or actions from within this
package, follow these steps:
* Let MSG_DEP_SET be the set of packages whose message types you use in
your messages/services/actions (e.g. std_msgs, actionlib_msgs, ...).
* In the file package.xml:
* add a build_depend tag for "message_generation"
* add a build_depend and a exec_depend tag for each package in MSG_DEP_SET
* If MSG_DEP_SET isn't empty the following dependency has been pulled in
but can be declared for certainty nonetheless:
* add a exec_depend tag for "message_runtime"
* In this file (CMakeLists.txt):
* add "message_generation" and every package in MSG_DEP_SET to
find_package(catkin REQUIRED COMPONENTS ...)
* add "message_runtime" and every package in MSG_DEP_SET to
catkin_package(CATKIN_DEPENDS ...)
* uncomment the add_*_files sections below as needed
and list every .msg/.srv/.action file to be processed
* uncomment the generate_messages entry below
* add every package in MSG_DEP_SET to generate_messages(DEPENDENCIES ...)
Generate messages in the 'msg' folder
Do you want to install the recommended 'CMake Tools' extension from Microsoft for CMakeLists.txt?
Install
Show Recommendations
ROS1.noetic
Ln 1, Col 1
Spaces: 2
UTF-8
LF
CMake

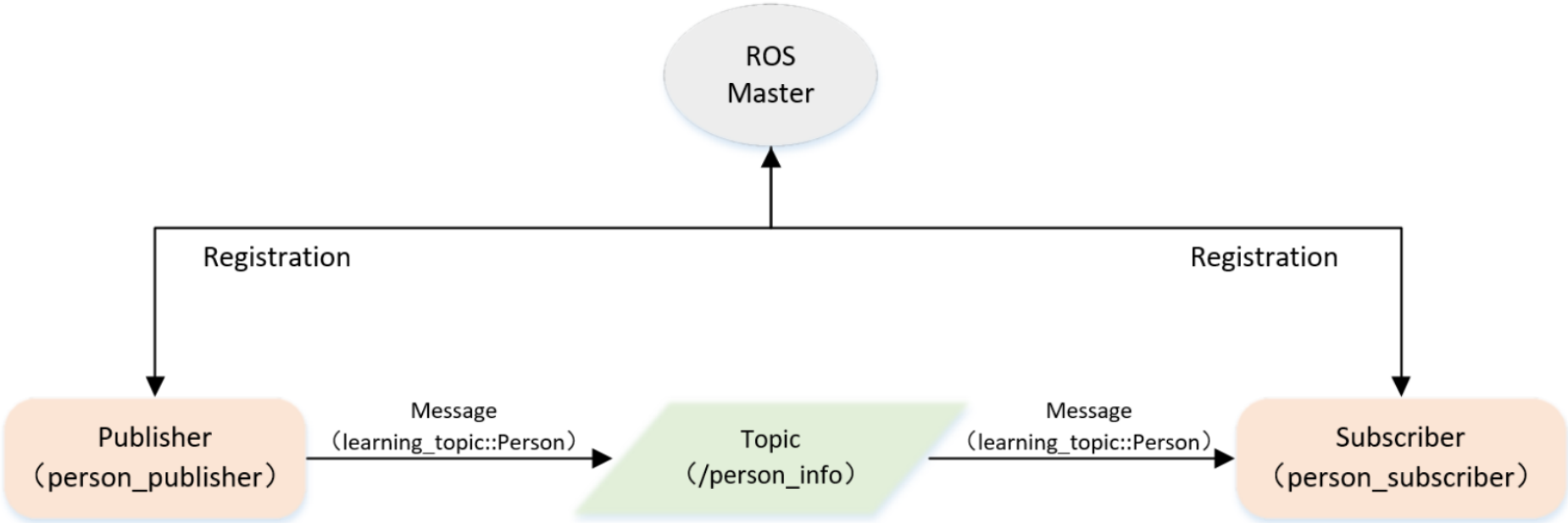
Scripts


Home
catkin_ws
src
learning_topic
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
include
msg
my_scripts
scripts
src
CMakeLists
.txt
package.
xml
ab@ab-c: ~/catkin_ws/src/learning_topic/scripts
ab@ab-c:~/catkin_ws/src/learning_topic$ mkdir scripts
ab@ab-c:~/catkin_ws/src/learning_topic$ cd scripts
ab@ab-c:~/catkin_ws/src/learning_topic/scripts$ touch velocity_publisher.py
ab@ab-c:~/catkin_ws/src/learning_topic/scripts$ chmod +x velocity_publisher.py
ab@ab-c:~/catkin_ws/src/learning_topic/scripts$


Activities
Visual Studio Code
Sep 26 00:54
Home
catkin_ws
src
Recent
Starred
Home
Desktop
Documer
Downloa
Music
Pictures
Videos
Trash
Other Lo
ab@ab-c: ~/catkin_ws/src
ab@ab-c:~/catkin_ws/src$ code .
ab@ab-c:~/catkin_ws/src$
my_forst_node.cpp - src - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
SRC
.vscode
learning_topic
my_scripts
test_pkg
CMakeLists.txt
OUTLINE
TIMELINE
my_forst_node.cpp
my_python > scripts > my_forst_node.cpp > ...
#include <ros/ros.h>
#include <geometry_msgs/Twist.h>
int main(int argc, char **argv)
{
ros::init(argc, argv, "velocity_publisher");
ros::NodeHandle n;
ros::Publisher turtle_vel_pub = n.advertise<geometry_msgs::Twis
ros::Rate loop_rate(10);
int count = 0;
while (ros::ok())
{
geometry_msgs::Twist vel_msg;
vel_msg.linear.x = 0.5;
vel_msg.angular.z = 0.2;
turtle_vel_pub.publish(vel_msg);
ROS_INFO("Publsh turtle velocity command[%0.2f m/s, %0.2f
vel_msg.linear.x, vel_msg.angular.z);
loop_rate.slee
}
return 0;
Do you want to install the recommended 'C/C++ Extension Pack' extension from Microsoft for the C++ language?
Install
Show Recommendations
ROS1.noetic
Ln 37, Col 1
Tab Size: 4
UTF-8
LF
C++
ROS

Lecture 7: Topic Message

Topic Model


ROS
Master
Registration
Registration
Publisher
（person_publisher）
Message
（learning_topic::Person）
Topic
（/person_info）
Message
（learning_topic::Person）
Subscriber
（person_subscriber）

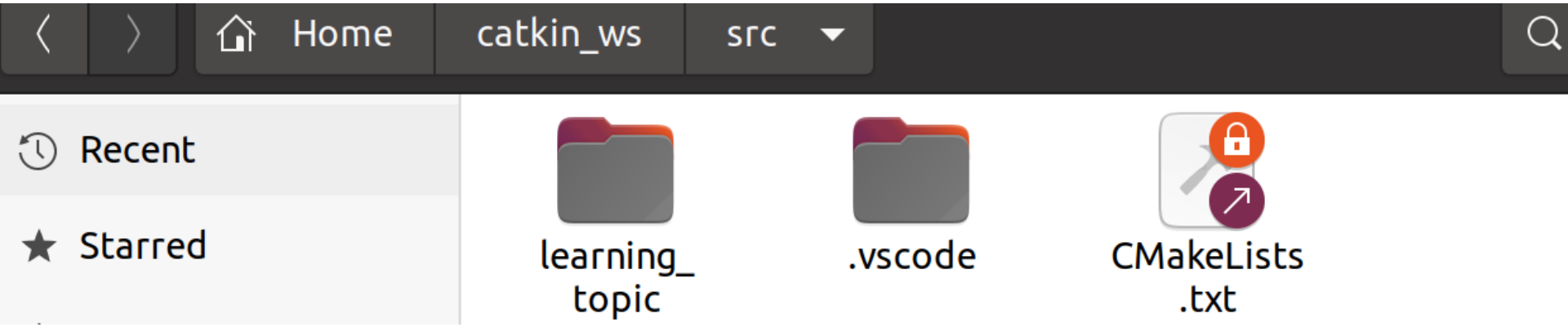
Topic Model

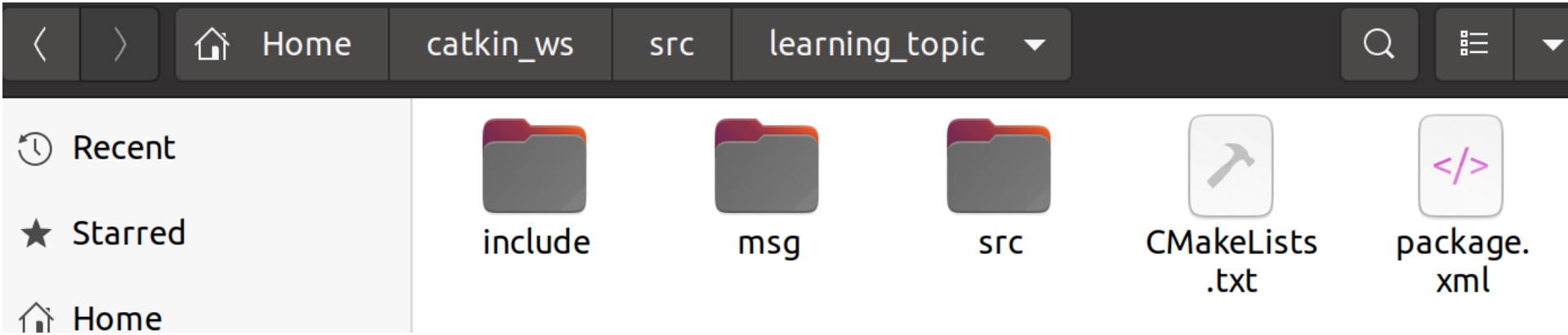
string name
uint8 sex
uint8 age

unit8 unknown = 0
uint8 male = 1
uint8 female =2

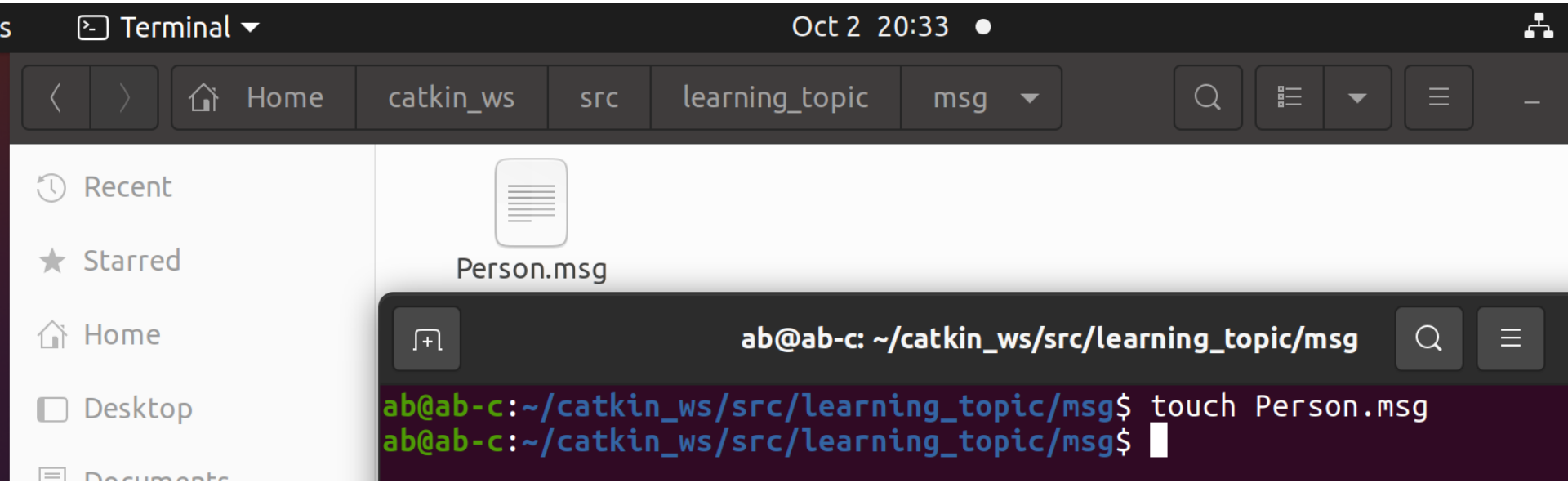
Person.msg

Build msg in learning_topic

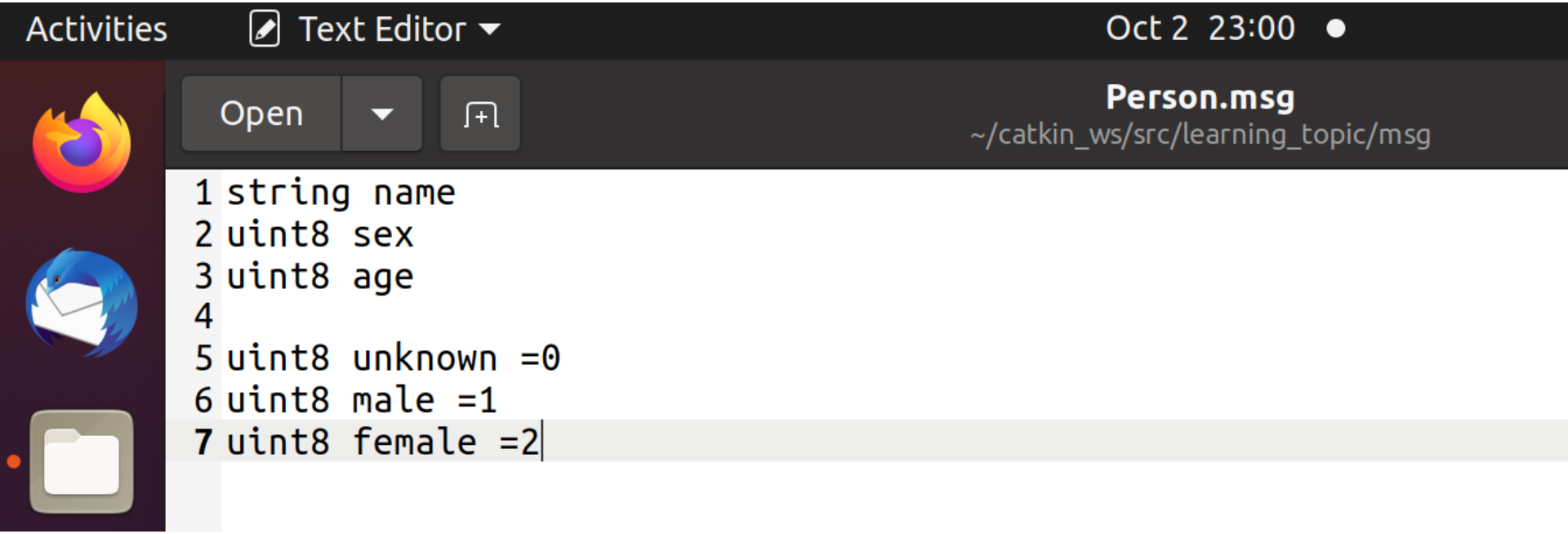




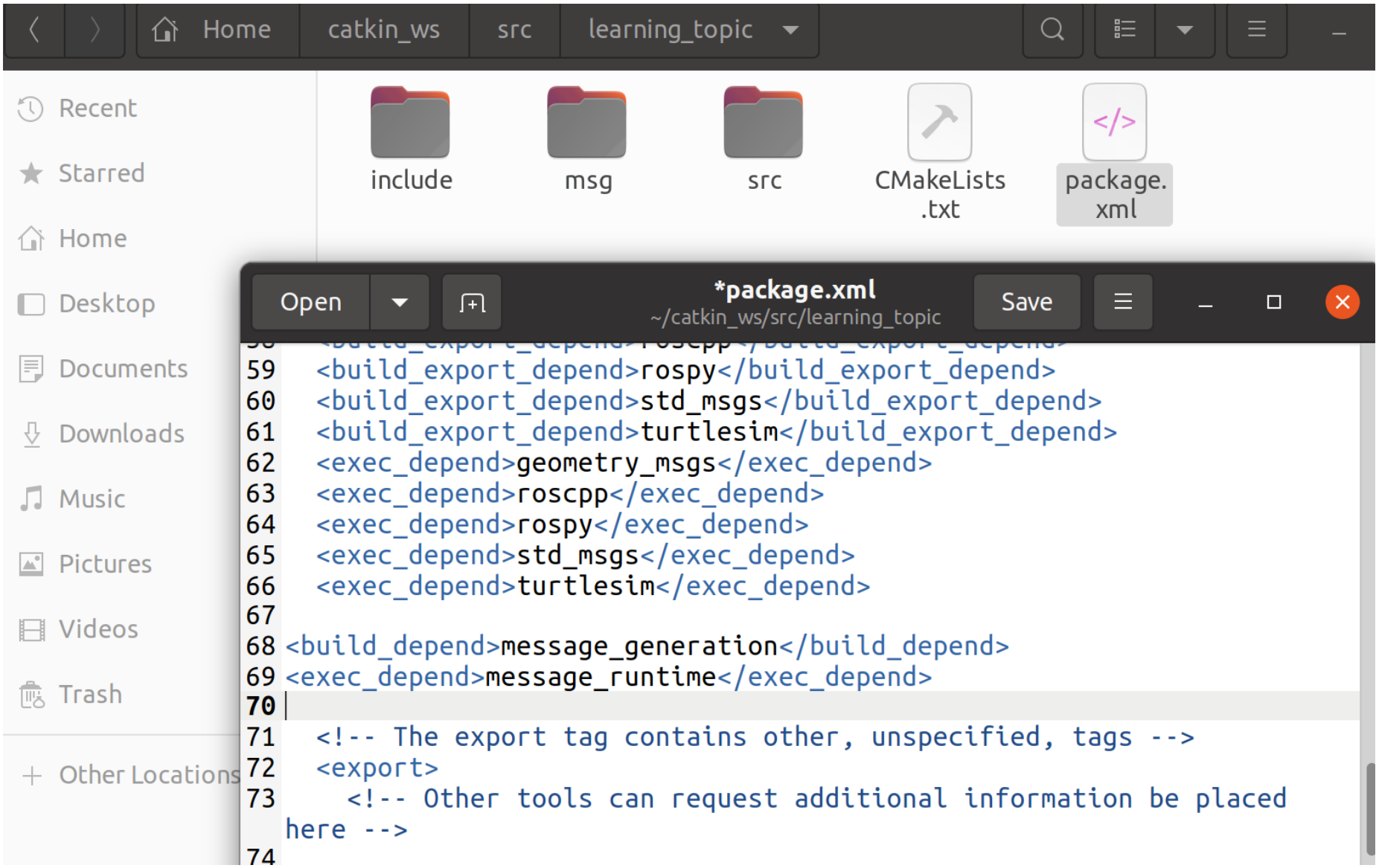
Build Person.msg



Data Interface Definition Person.msg

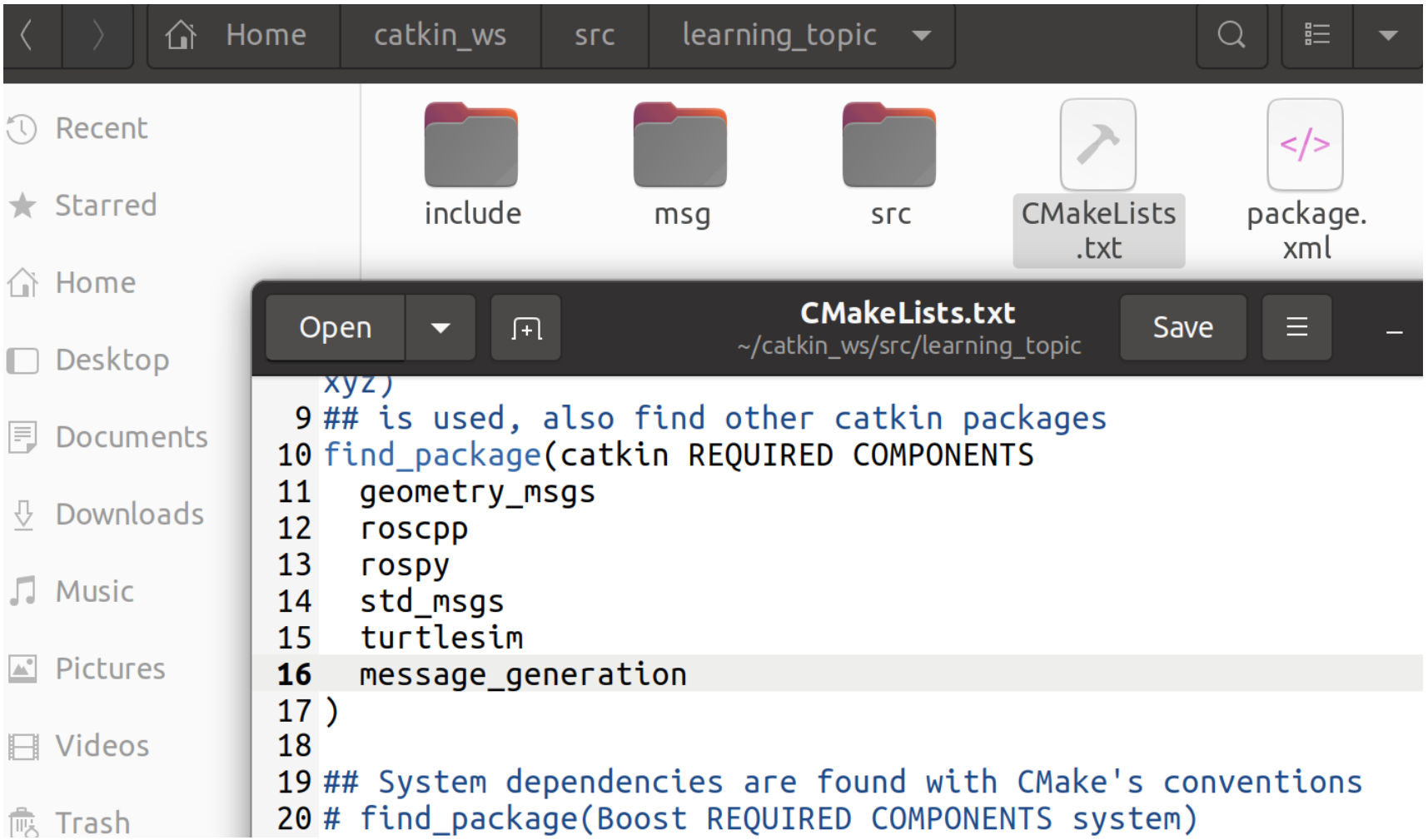


Add Dependence in package.xml



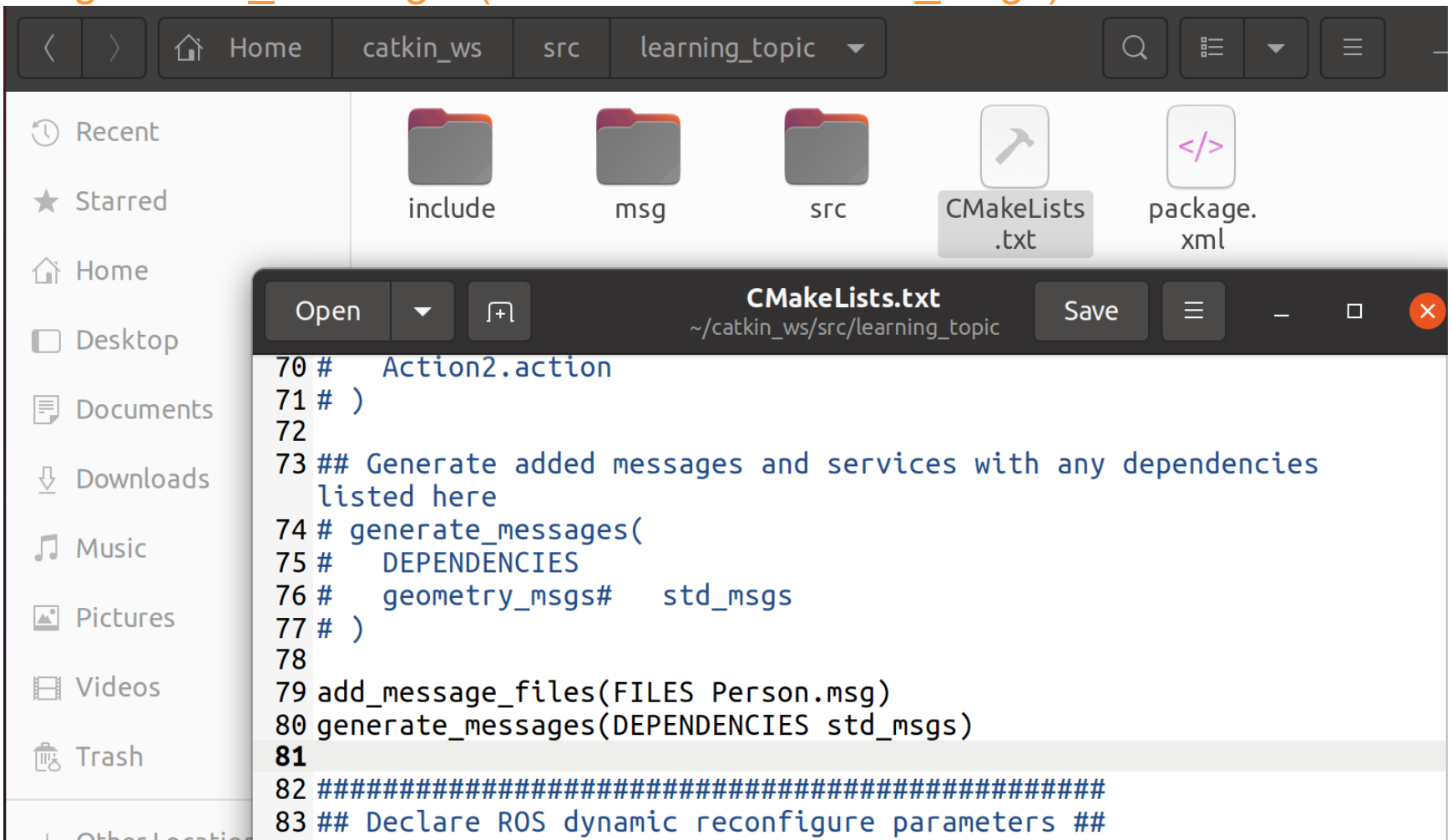
Add Compile Options in CMakeLists.txt

1. message_generation



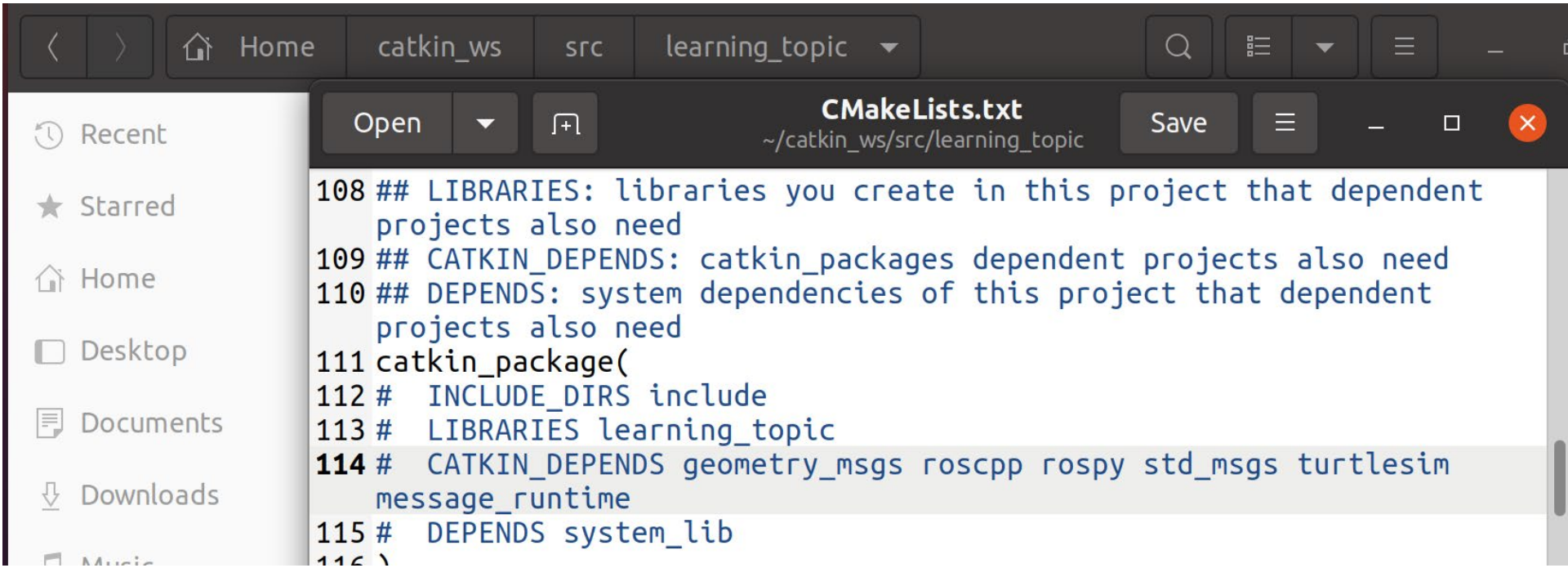
Add Compile Options in CMakeLists.txt

2. add_message_files(FILES Person.msg)
generate_messages(DEPENDENCIES std_msgs)

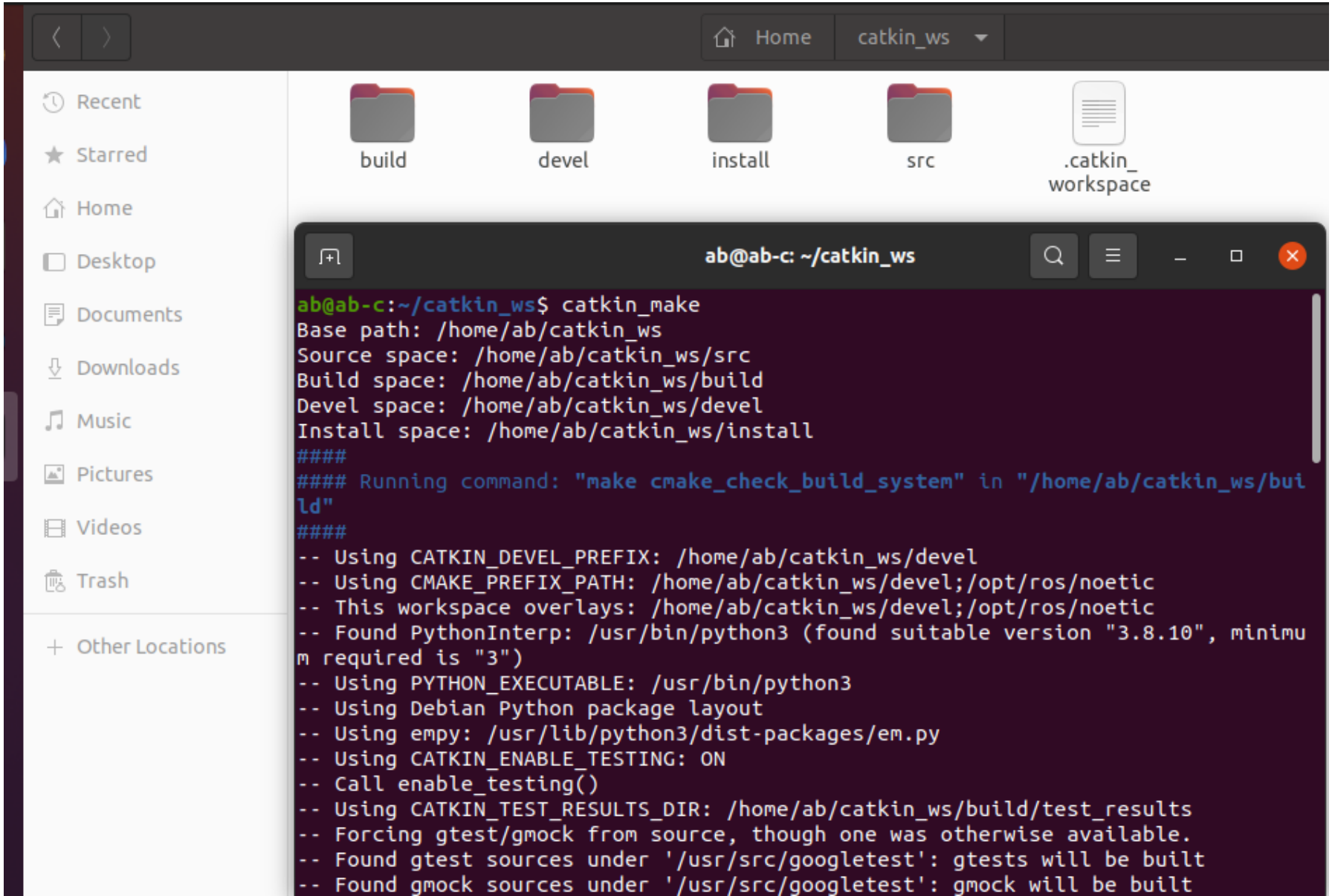


Add Compile Options in CMakeLists.txt

3. message_runtime



catkin_make Compile

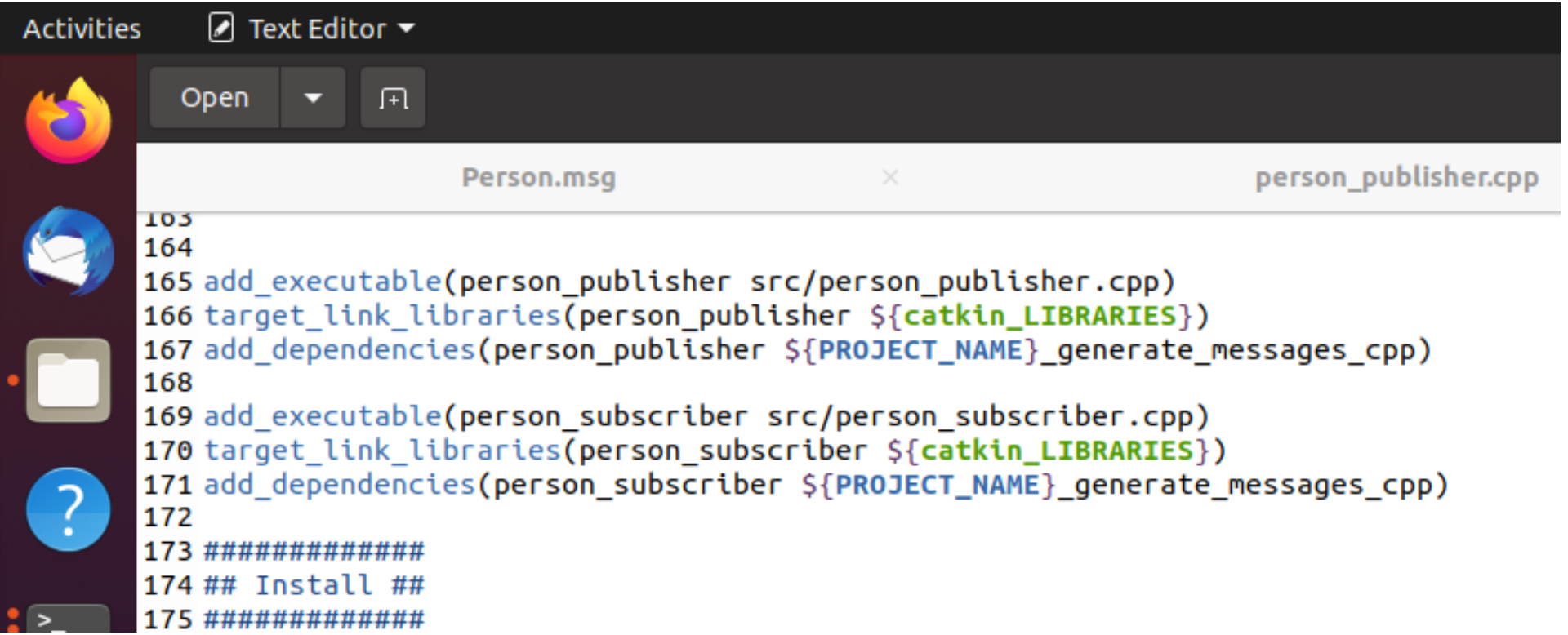


Create Publisher Code C++

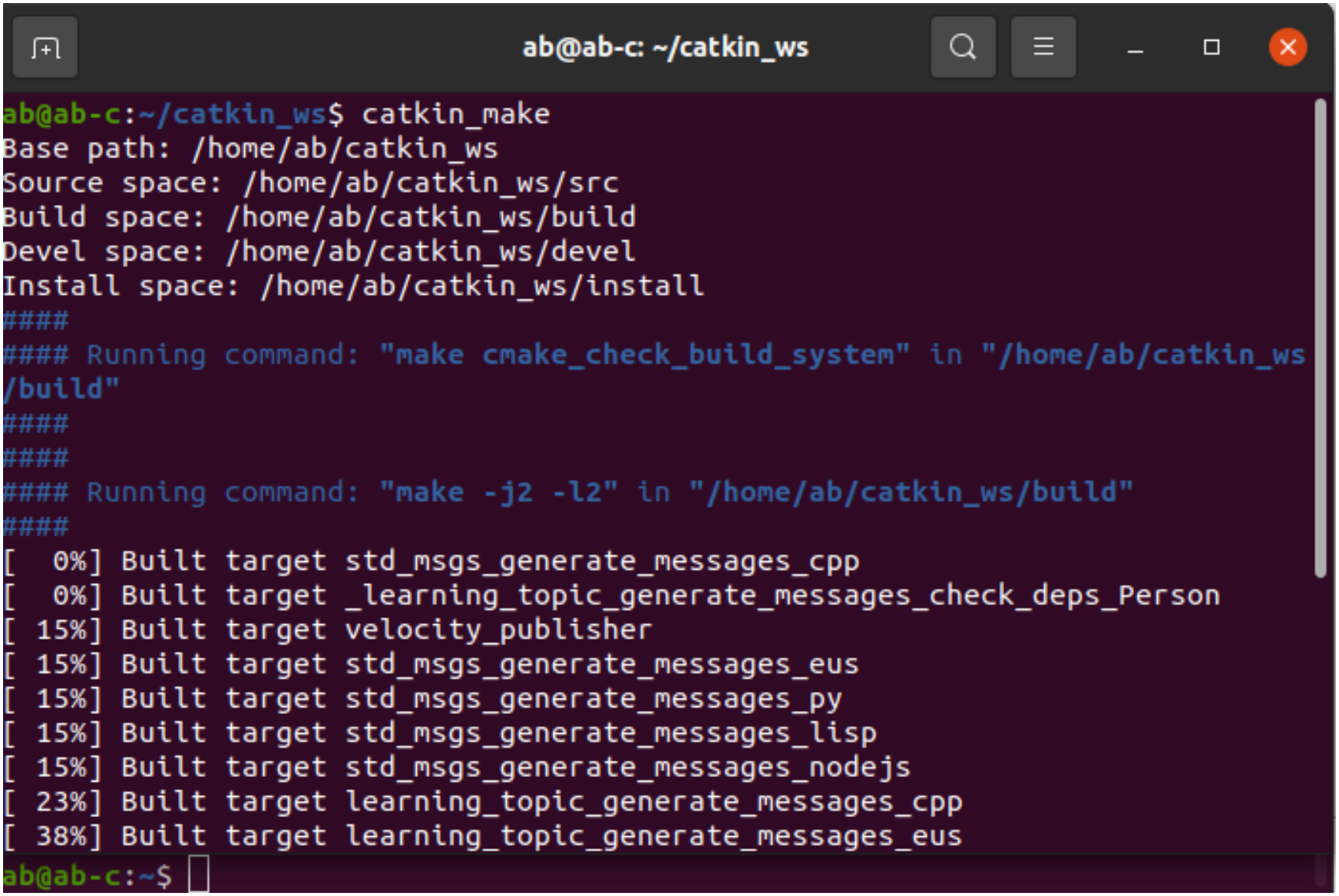
Person.msg

```cpp
#include <ros/ros.h>
#include "learning_topic/Person.h"

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

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


    ros::NodeHandle n;


    ros::Publisher person_info_pub = n.advertise<learning_topic::Person>("/person_info", 10);


    ros::Rate loop_rate(1);

    int count = 0;
    while (ros::ok())
    {

        learning_topic::Person person_msg;
                person_msg.name = "Tom";
                person_msg.age  = 18;
                person_msg.sex  = learning_topic::Person::male;


                person_info_pub.publish(person_msg);

                ROS_INFO("Publish Person Info: name:%s  age:%d  sex:%d",
                                person_msg.name.c_str(), person_msg.age, person_msg.sex);


        loop_rate.sleep();
    }

    return 0;
}
```

Compiling Code



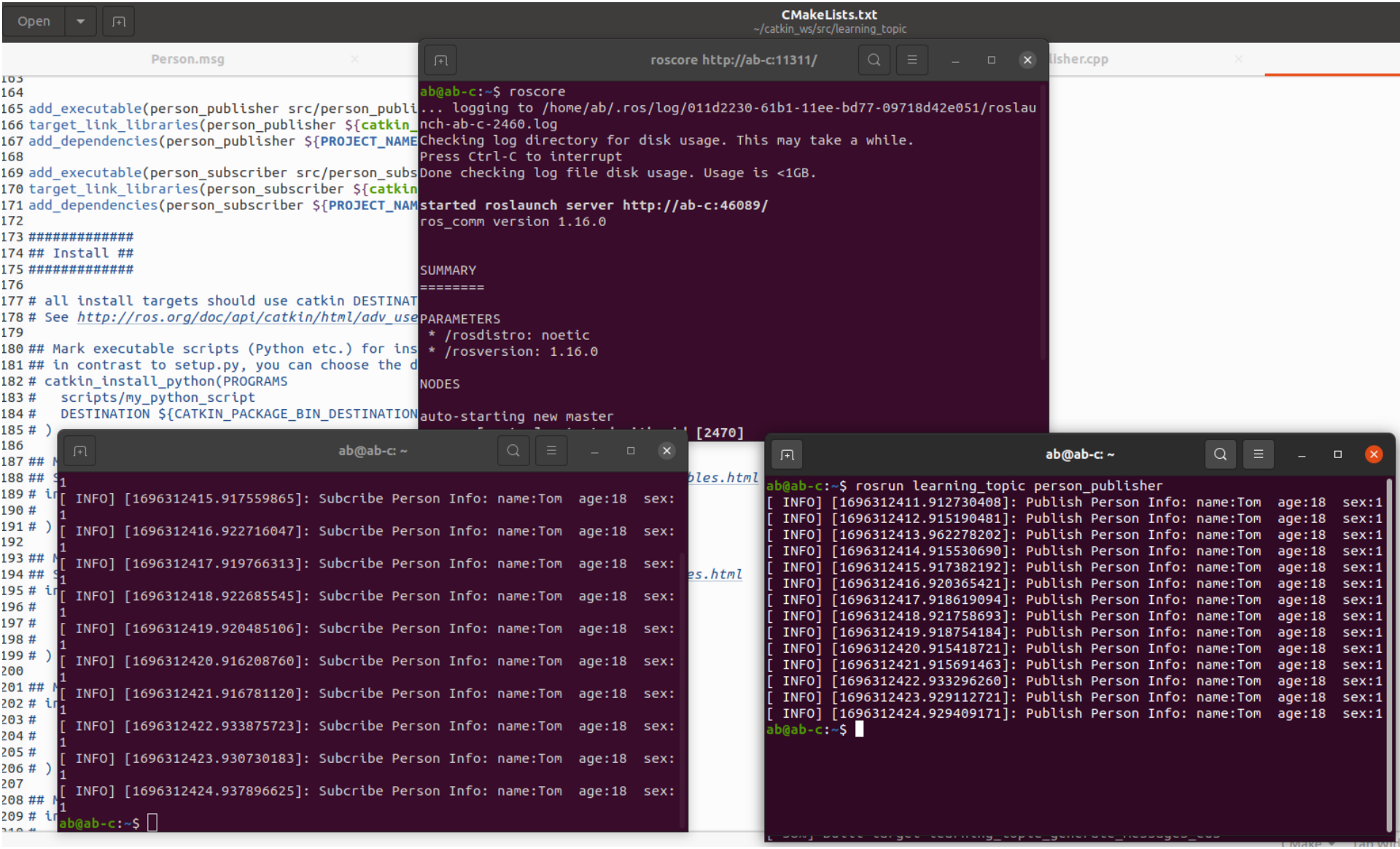
catkin_make Compile



Publisher and Subscriber Compile

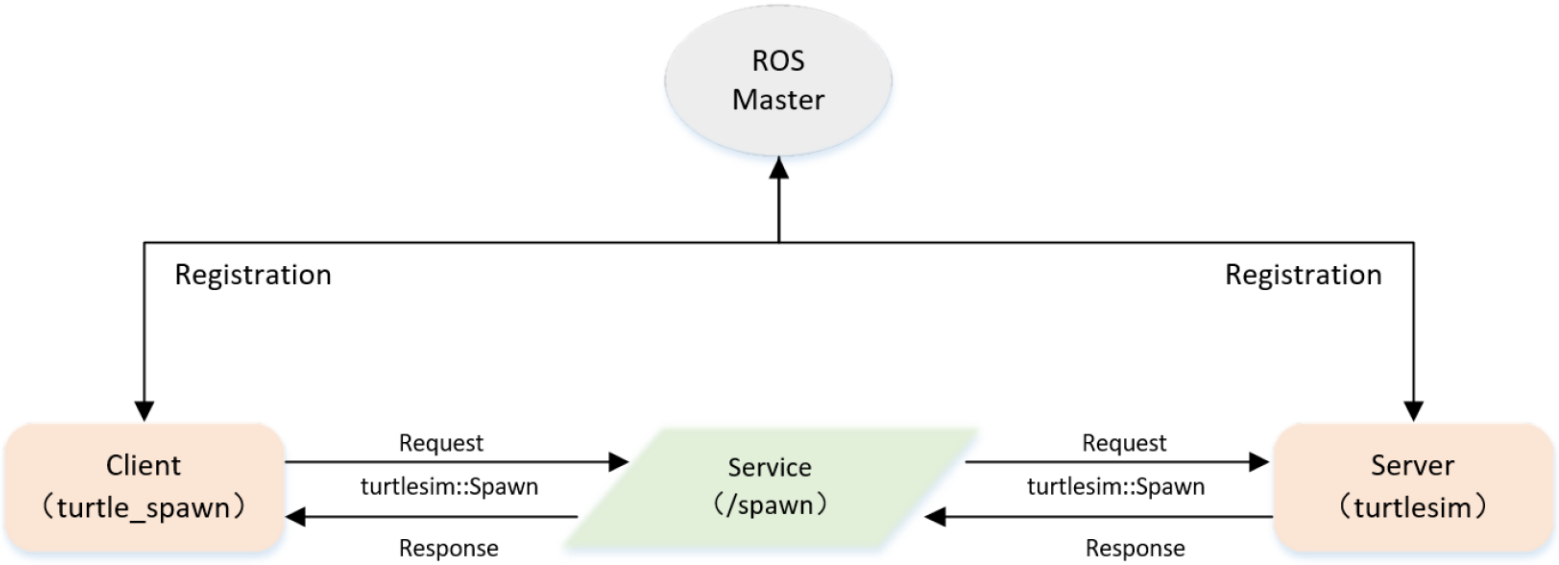
```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun learning_topic person_subscriber
$ rosrun learning_topic person_publisher
```

Publisher and Subscriber Compile



Lecture 8: Client Program

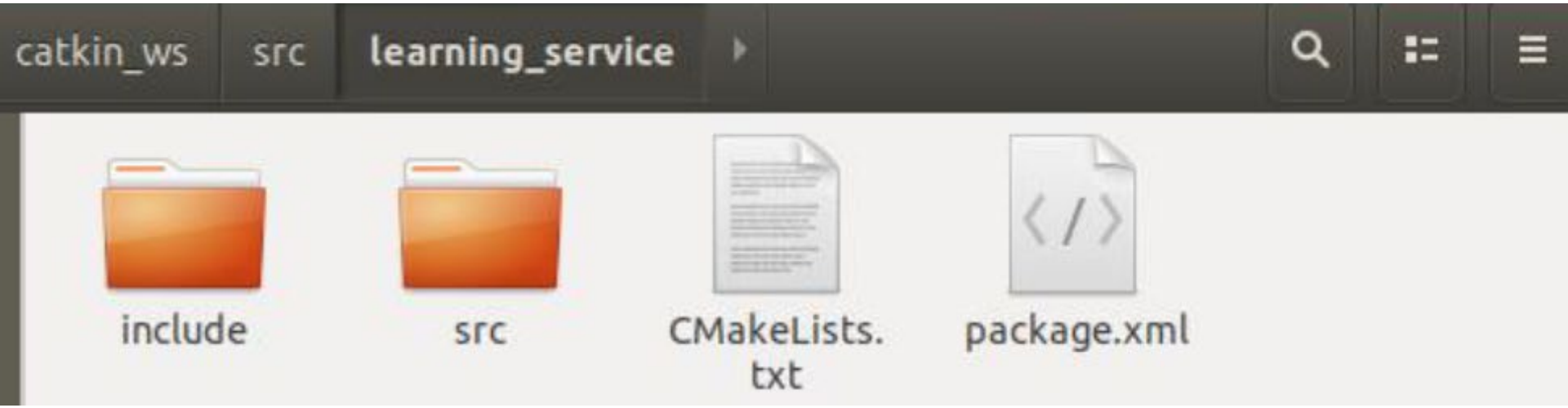
Publisher and Subscriber Compile



Build Package

$ cd ~/catkin_ws/src

$ catkin_create_pkg learning_service roscpp rospy std_msgs geometry_msgs turtlesim



C++ Code

Open

```cpp
#include <ros/ros.h>
#include <turtlesim/Spawn.h>

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

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


        ros::NodeHandle node;


        ros::service::waitForService("/spawn");
        ros::ServiceClient add_turtle = node.serviceClient<turtlesim::Spawn>("/spawn");


        turtlesim::Spawn srv;
        srv.request.x = 2.0;
        srv.request.y = 2.0;
        srv.request.name = "turtle2";


        ROS_INFO("Call service to spwan turtle[x:%0.6f, y:%0.6f, name:%s]",
                        srv.request.x, srv.request.y, srv.request.name.c_str());

        add_turtle.call(srv);

        ROS_INFO("Spwan turtle successfully [name:%s]", srv.response.name.c_str());

        return 0;
};
```

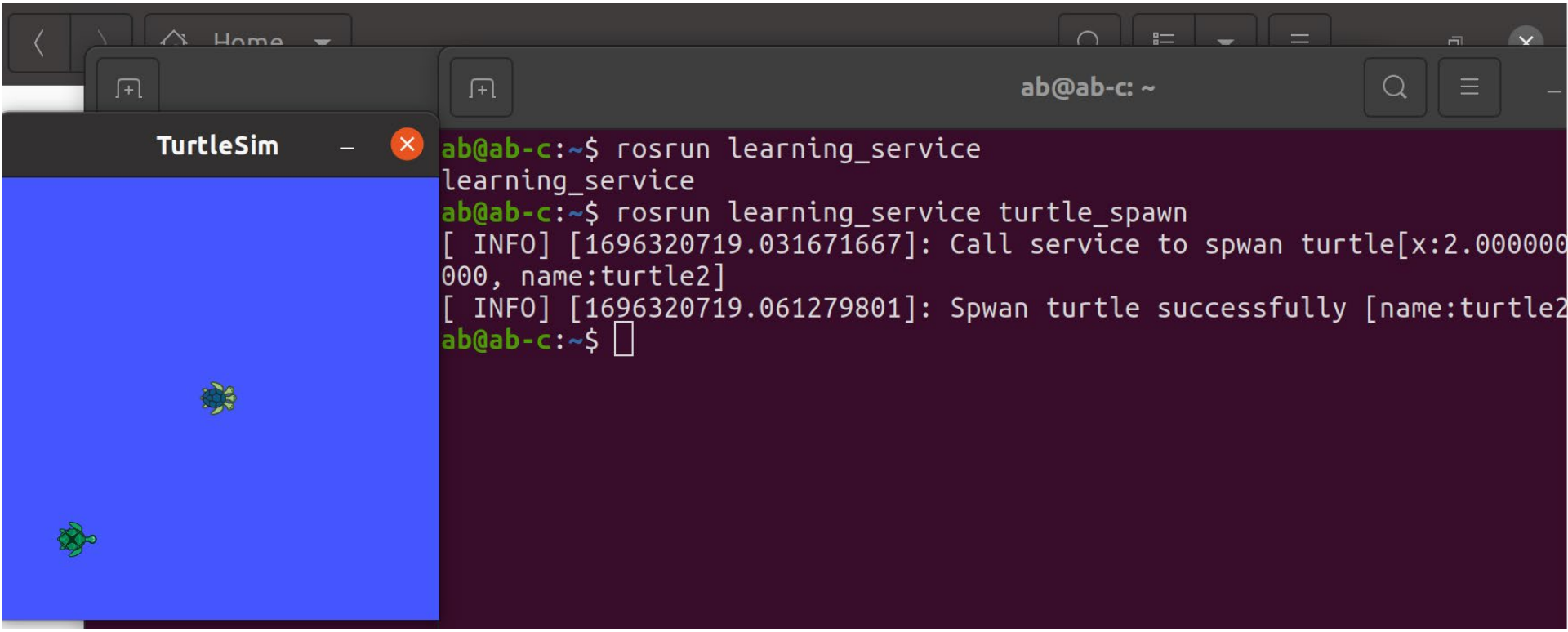
Compiling Code

```
## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/learning_service_node.cpp)

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

add_executable(turtle_spawn src/turtle_spawn.cpp)
target_link_libraries(turtle_spawn ${catkin_LIBRARIES})
```

Compiling Code

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun turtlesim turtlesim_node
$ rosrun learning_service turtle_spawn
```

Compiling Code



Python Code

Open

```
1 import sys
2 import rospy
3 from turtlesim.srv import Spawn
4
5 def turtle_spawn():
6
7     rospy.init_node('turtle_spawn')
8
9
10    rospy.wait_for_service('/spawn')
11    try:
12        add_turtle = rospy.ServiceProxy('/spawn', Spawn)
13
14
15        response = add_turtle(2.0, 2.0, 0.0, "turtle2")
16        return response.name
17    except rospy.ServiceException, e:
18        print "Service call failed: %s"%e
19
20 if __name__ == "__main__":
21
22    print "Spwan turtle successfully [name:%s]" %(turtle_spawn())
23
24
```

Reference

✓ Download link

✓ Language:

English, Chinese, Japanese, Korean

"ROS Robot Programming"

A Handbook is written by TurtleBot3 Developers

Reference

- R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza. Introduction to Autonomous Mobile Robots. MIT Press, 2nd Edition, 2011, ISBN-10: 0262015358.
- Y. Pyo, H. Cho, R. Jung, and T. Lim, ROS Robot Programming, ROBOTIS Co., Ltd., 2017, ISBN 979-11-962307-1-5
- J. O'Kane, A Gentle Introduction to ROS, CreateSpace Independent Publishing Platform, 2013, ISBN-13: 978-1492143239