

ECE 4703

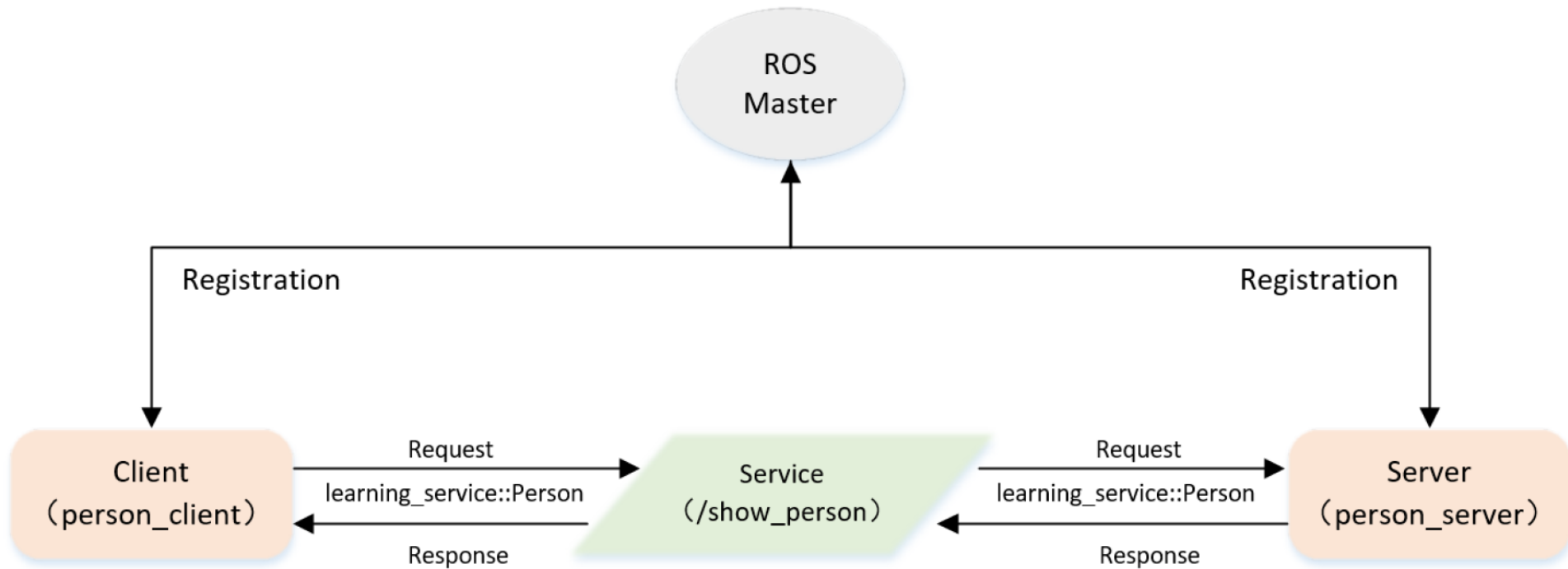
Mobile Autonomous Robots

Jenny Zhen Yu
zhenyu@cpp.edu

Department of Electrical and Computer Engineering
California State Polytechnic University, Pomona

Lecture 10: Service Data

Server Model



Service File

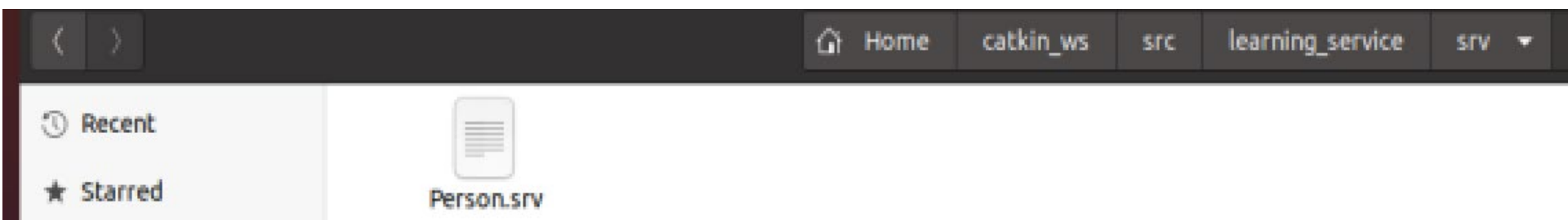
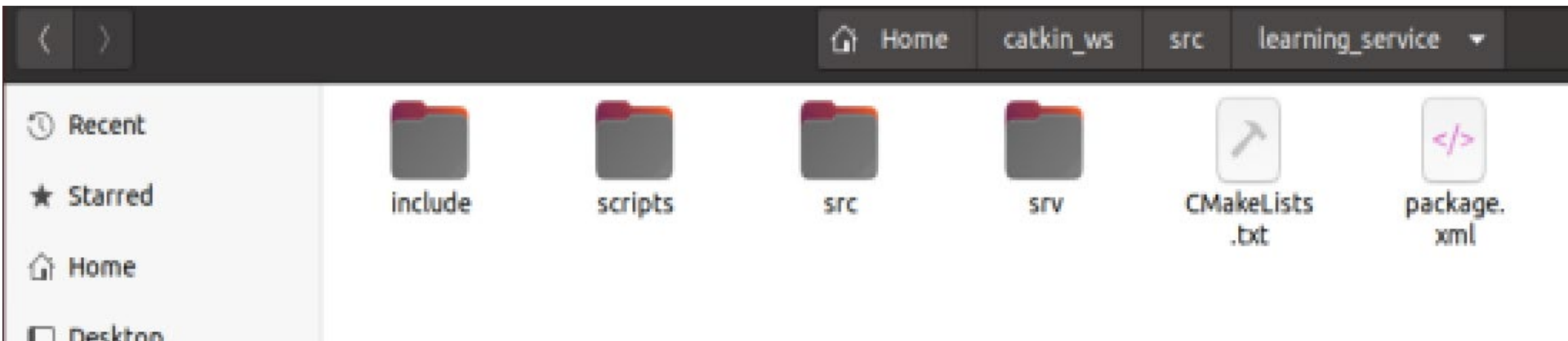
string name
uint8 sex
uint8 age

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

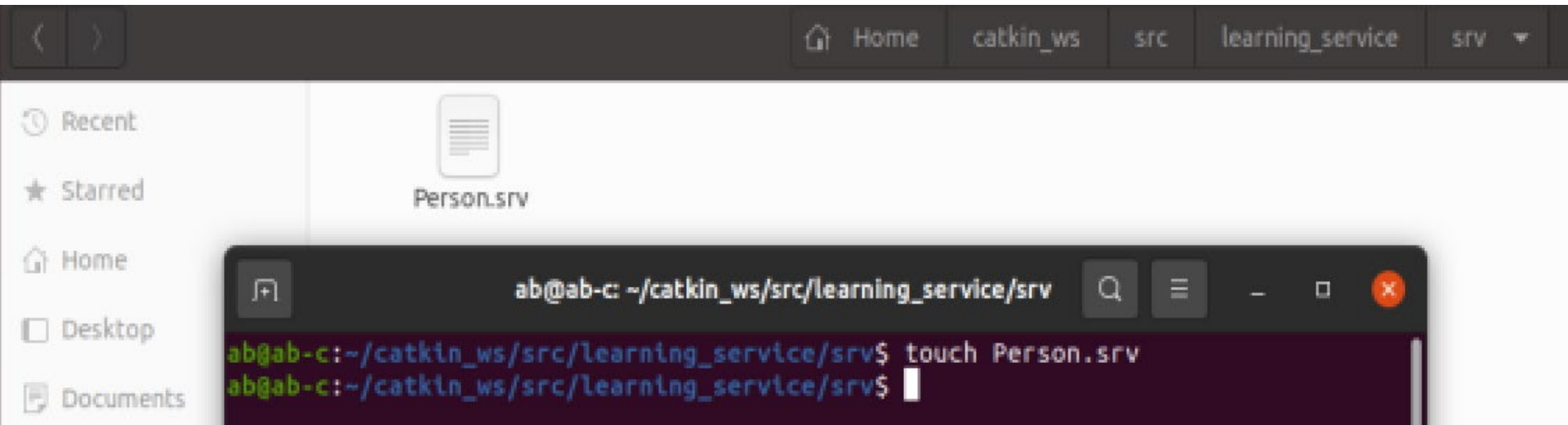
string result

touch Person.srv

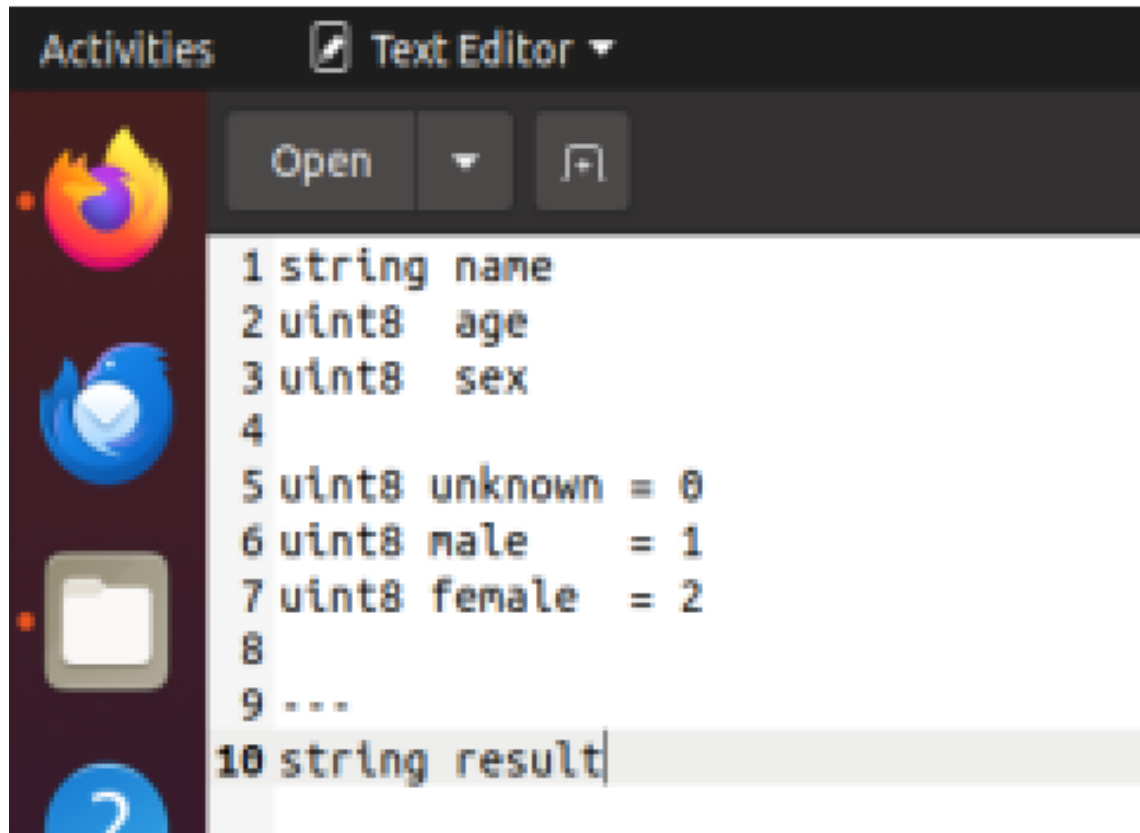
Build srv in learning_service



Build Service File Person.srv



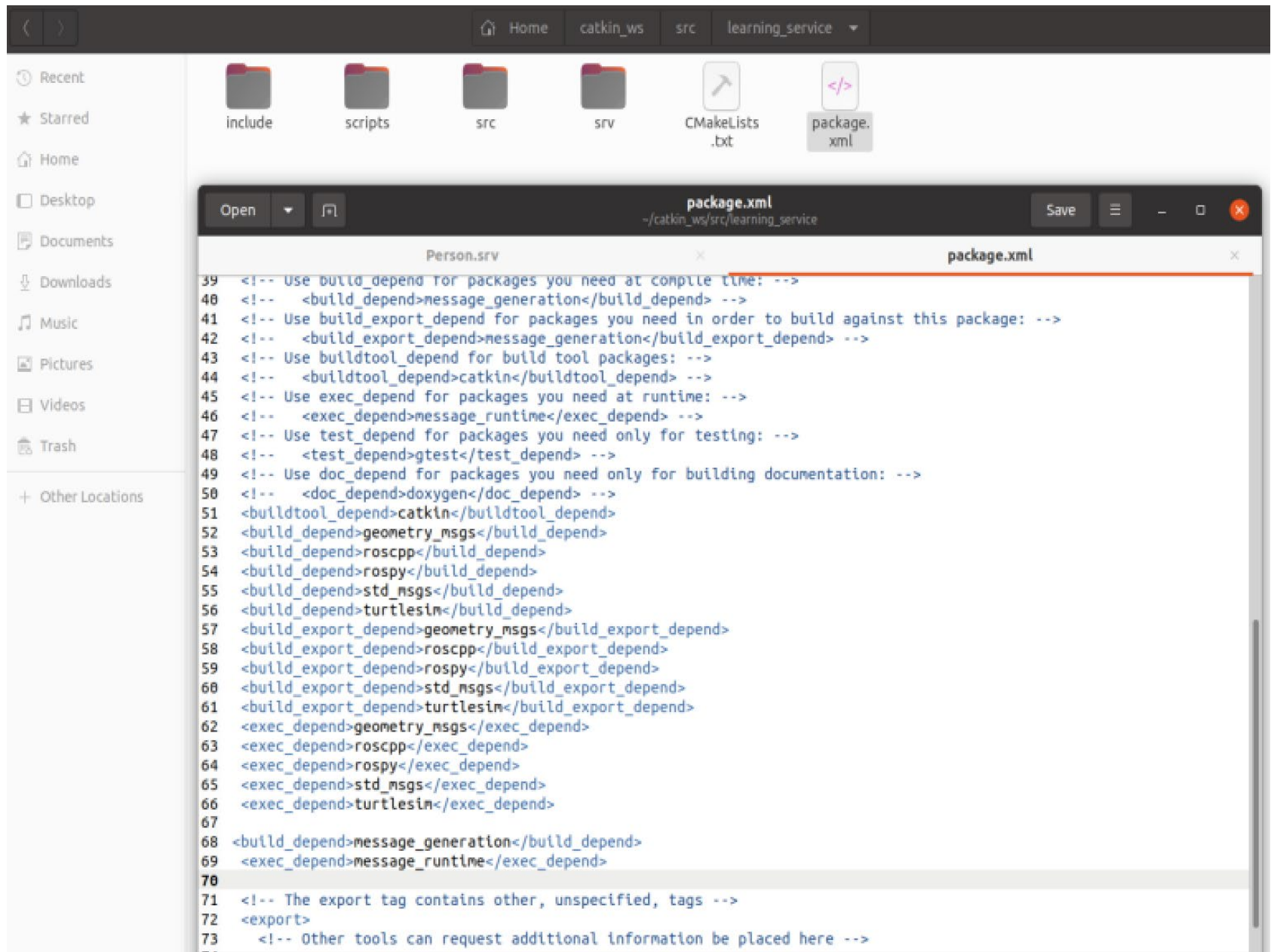
Data Interface Definition Person.srv



The screenshot shows a Linux desktop environment. On the left is a vertical dock with icons for Firefox, a mail client, a file manager, and a terminal. The main window is a text editor titled 'Text Editor'. It contains a list of data fields for a service interface, numbered 1 through 10. The fields are: 1 string name, 2 uint8 age, 3 uint8 sex, 4, 5 uint8 unknown = 0, 6 uint8 male = 1, 7 uint8 female = 2, 8, 9 ..., and 10 string result. The text is in a monospaced font, and the line numbers are on the left of each field.

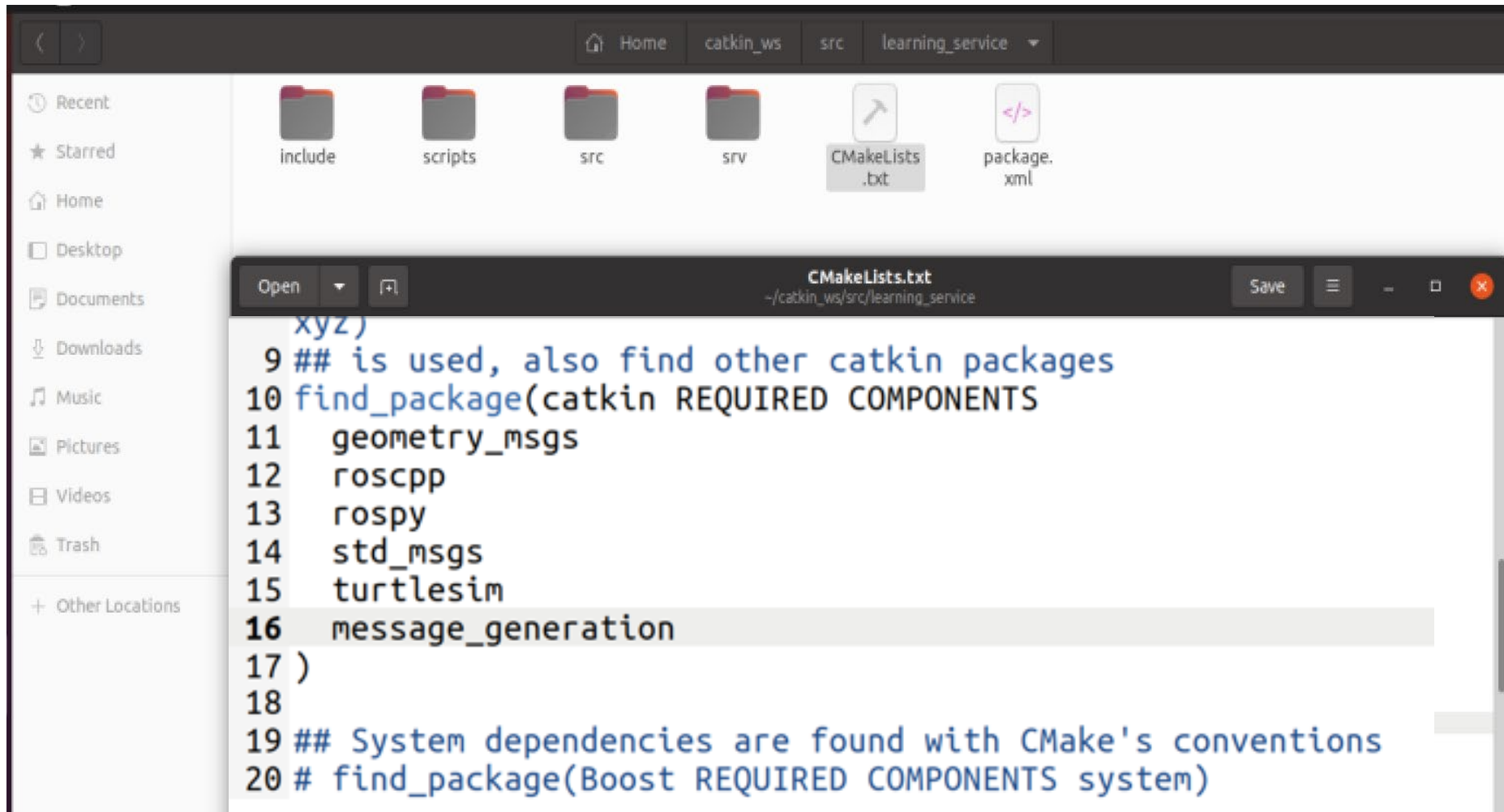
```
1 string name
2 uint8  age
3 uint8  sex
4
5 uint8 unknown = 0
6 uint8 male    = 1
7 uint8 female  = 2
8
9 ...
10 string result
```

Add Dependence in package.xml



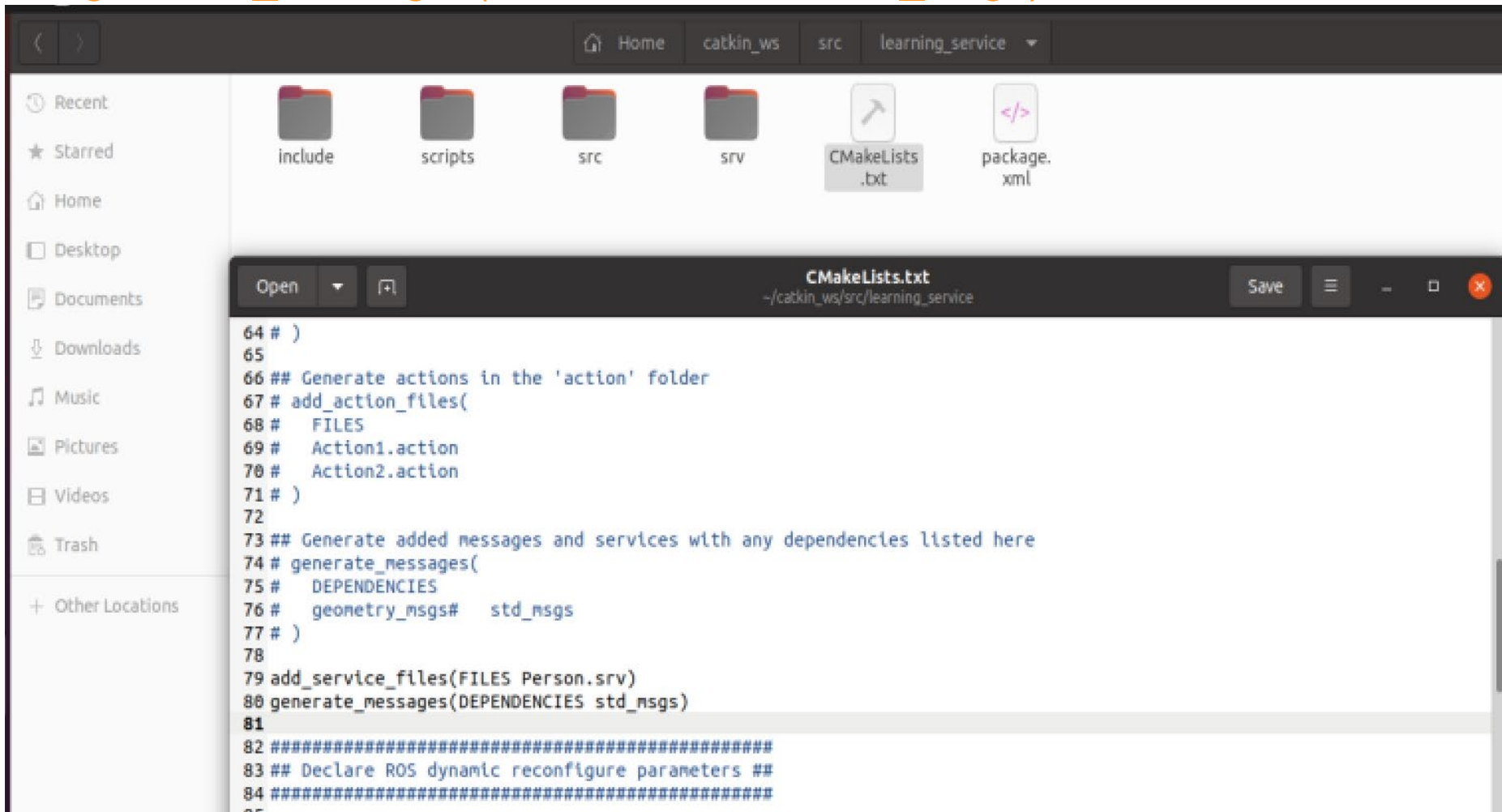
Add Compile Options in CMakeLists.txt

1. message_generation



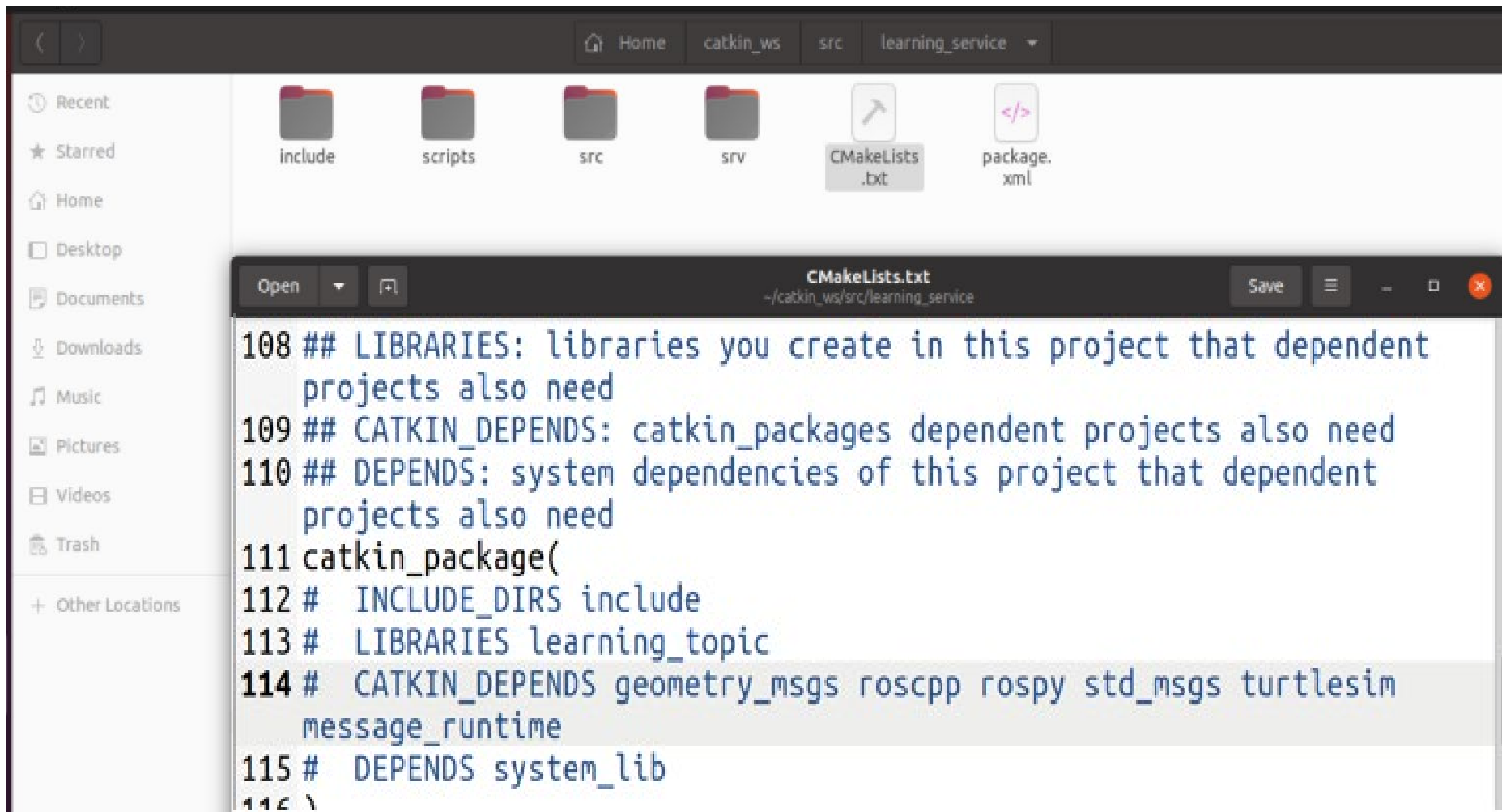
Add Compile Options in CMakeLists.txt

2. `add_message_files(FILES Person.srv)`
`generate_messages(DEPENDENCIES std_msgs)`

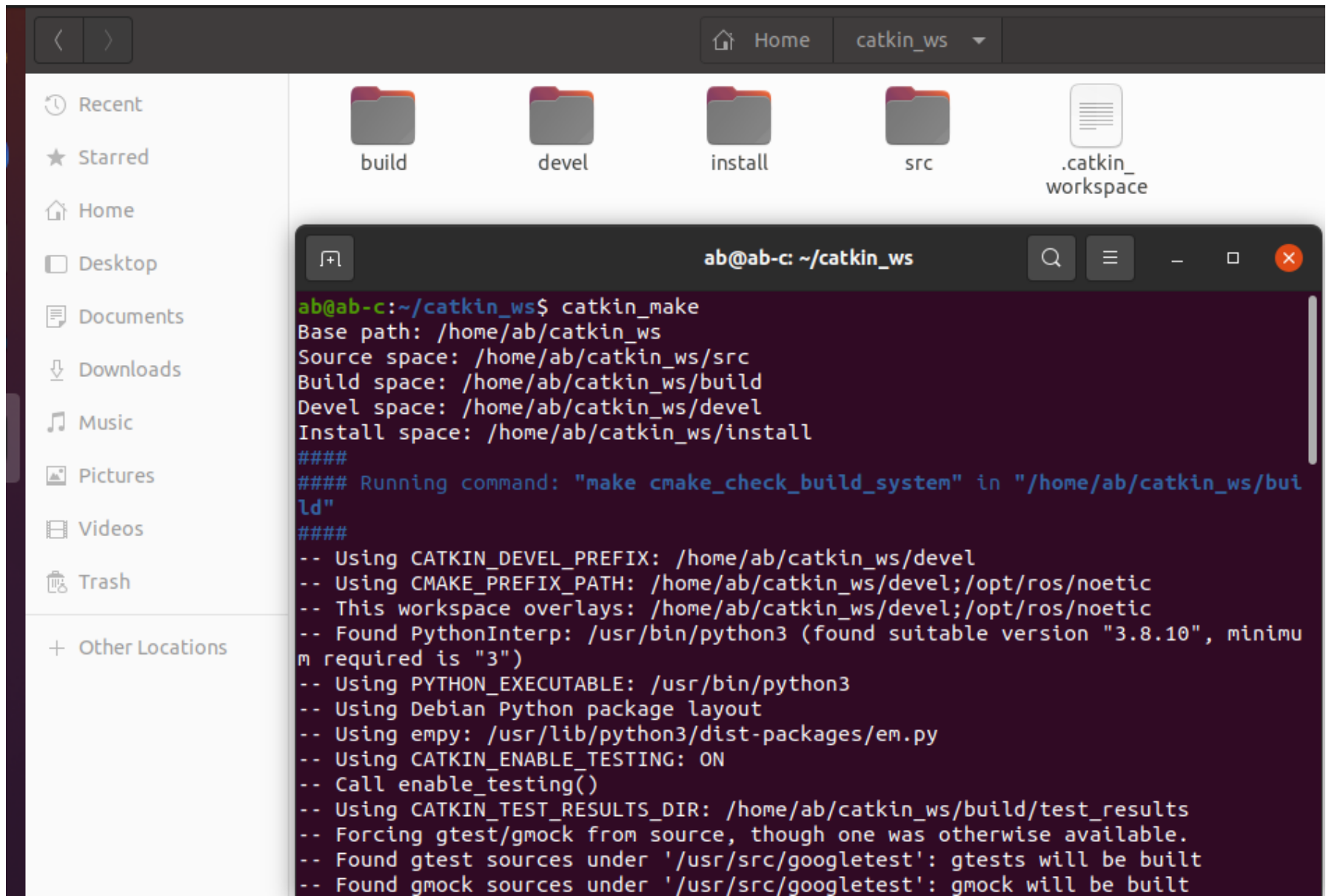


Add Compile Options in CMakeLists.txt

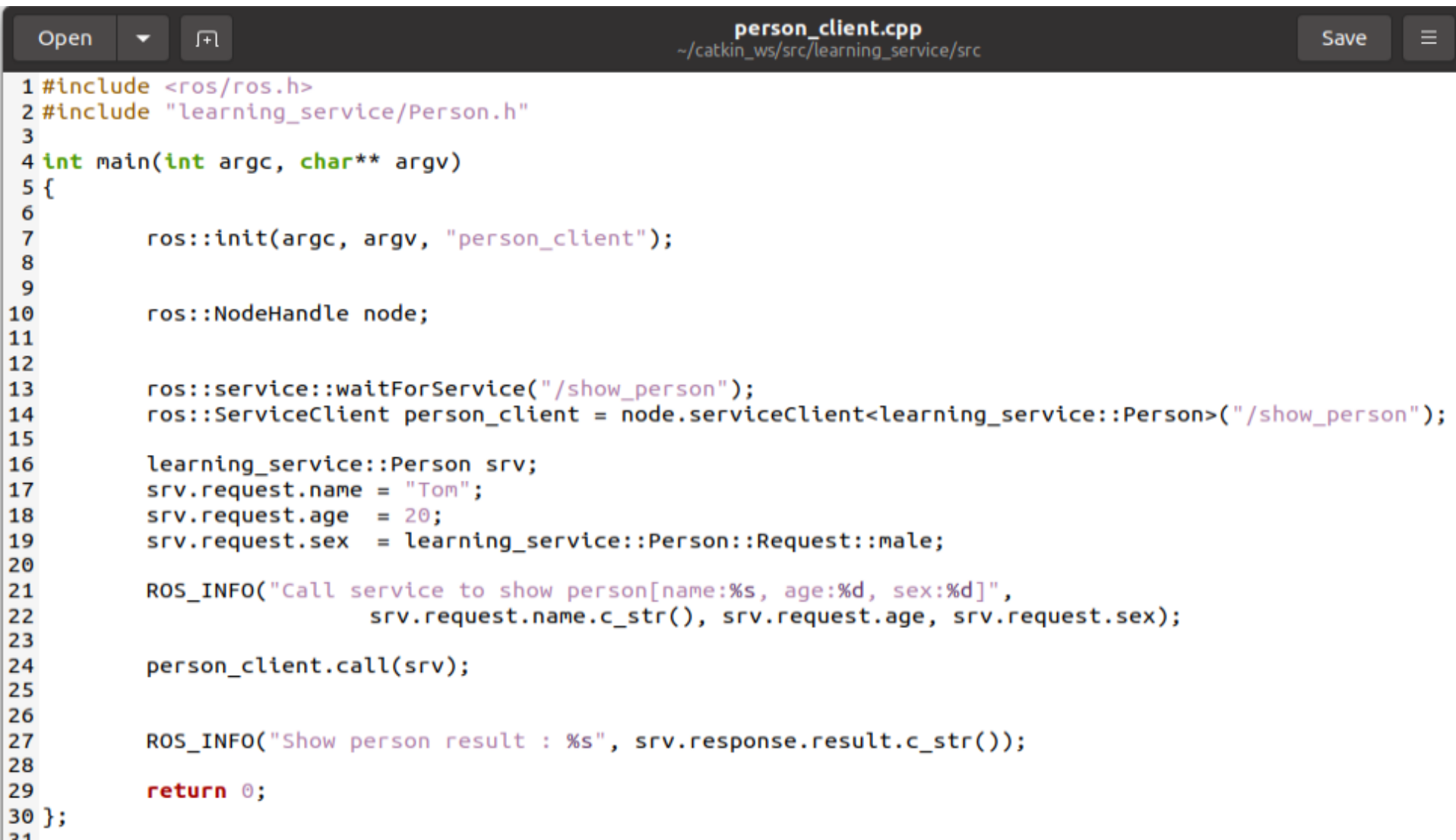
1. message_generation



catkin_make Compile



Create the Service Client Node Code C++



The image shows a code editor window with a dark theme. The title bar at the top reads "person_client.cpp" and the file path is "~/catkin_ws/src/learning_service/src". There are buttons for "Open", "Save", and a menu icon. The code is written in C++ and implements a ROS service client. It includes the ROS headers and the service definition header. The main function initializes the node, waits for the service, creates a service client, sets the request parameters (name: "Tom", age: 20, sex: male), calls the service, and prints the result. The code is line-numbered from 1 to 30.

```
1 #include <ros/ros.h>
2 #include "learning_service/Person.h"
3
4 int main(int argc, char** argv)
5 {
6     ros::init(argc, argv, "person_client");
7
8
9     ros::NodeHandle node;
10
11
12     ros::service::waitForService("/show_person");
13     ros::ServiceClient person_client = node.serviceClient<learning_service::Person>("/show_person");
14
15     learning_service::Person srv;
16     srv.request.name = "Tom";
17     srv.request.age = 20;
18     srv.request.sex = learning_service::Person::Request::male;
19
20     ROS_INFO("Call service to show person[name:%s, age:%d, sex:%d]",
21             srv.request.name.c_str(), srv.request.age, srv.request.sex);
22
23     person_client.call(srv);
24
25
26     ROS_INFO("Show person result : %s", srv.response.result.c_str());
27
28     return 0;
29 }
30
```

Create the Service Server Node Code C++

```
person_server.cpp
~/catkin_ws/src/learning_service/src

1 #include <ros/ros.h>
2 #include "learning_service/Person.h"
3
4
5 bool personCallback(learning_service::Person::Request &req,
6                     learning_service::Person::Response &res)
7 {
8
9     ROS_INFO("Person: name:%s age:%d sex:%d", req.name.c_str(), req.age, req.sex);
10
11     res.result = "OK";
12
13     return true;
14 }
15
16
17 int main(int argc, char **argv)
18 {
19
20     ros::init(argc, argv, "person_server");
21
22
23     ros::NodeHandle n;
24
25
26     ros::ServiceServer person_service = n.advertiseService("/show_person", personCallback);
27
28
29     ROS_INFO("Ready to show person informtion.");
30     ros::spin();
31
32     return 0;
33 }
```


Compiling Code

The screenshot shows a file manager interface with a sidebar on the left containing navigation options: Recent, Starred, Home, Desktop, Documents, Downloads, Music, Pictures, Videos, Trash, and Other Locations. The main pane displays the contents of the 'learning_service' directory, including folders 'include', 'scripts', 'src', and 'srv', and files 'CMakeLists.txt' (highlighted) and 'package.xml'. Below this, a window titled 'CMakeLists.txt' is open, showing the file path '~/catkin_ws/src/learning_service'. The code in the window is for 'person_server.cpp' and includes the following CMake commands:

```
155 # target_link_libraries(${PROJECT_NAME}_node
156 #   ${catkin_LIBRARIES})
157 # )
158
159 add_executable(turtle_spawn src/turtle_spawn.cpp)
160 target_link_libraries(turtle_spawn ${catkin_LIBRARIES})
161
162 add_executable(turtle_command_server src/turtle_command_server.cpp)
163 target_link_libraries(turtle_command_server ${catkin_LIBRARIES})
164
165 # add_executable(turtle_command_server scripts/turtle_command_server.py)
166 # target_link_libraries(turtle_command_server ${catkin_LIBRARIES})
167
168 add_executable(person_server src/person_server.cpp)
169 target_link_libraries(person_server ${catkin_LIBRARIES})
170 add_dependencies(person_server ${PROJECT_NAME}_gencpp)
171
172 add_executable(person_client src/person_client.cpp)
173 target_link_libraries(person_client ${catkin_LIBRARIES})
174 add_dependencies(person_client ${PROJECT_NAME}_gencpp)
175
176 #####
177 ## Install ##
178 #####
```

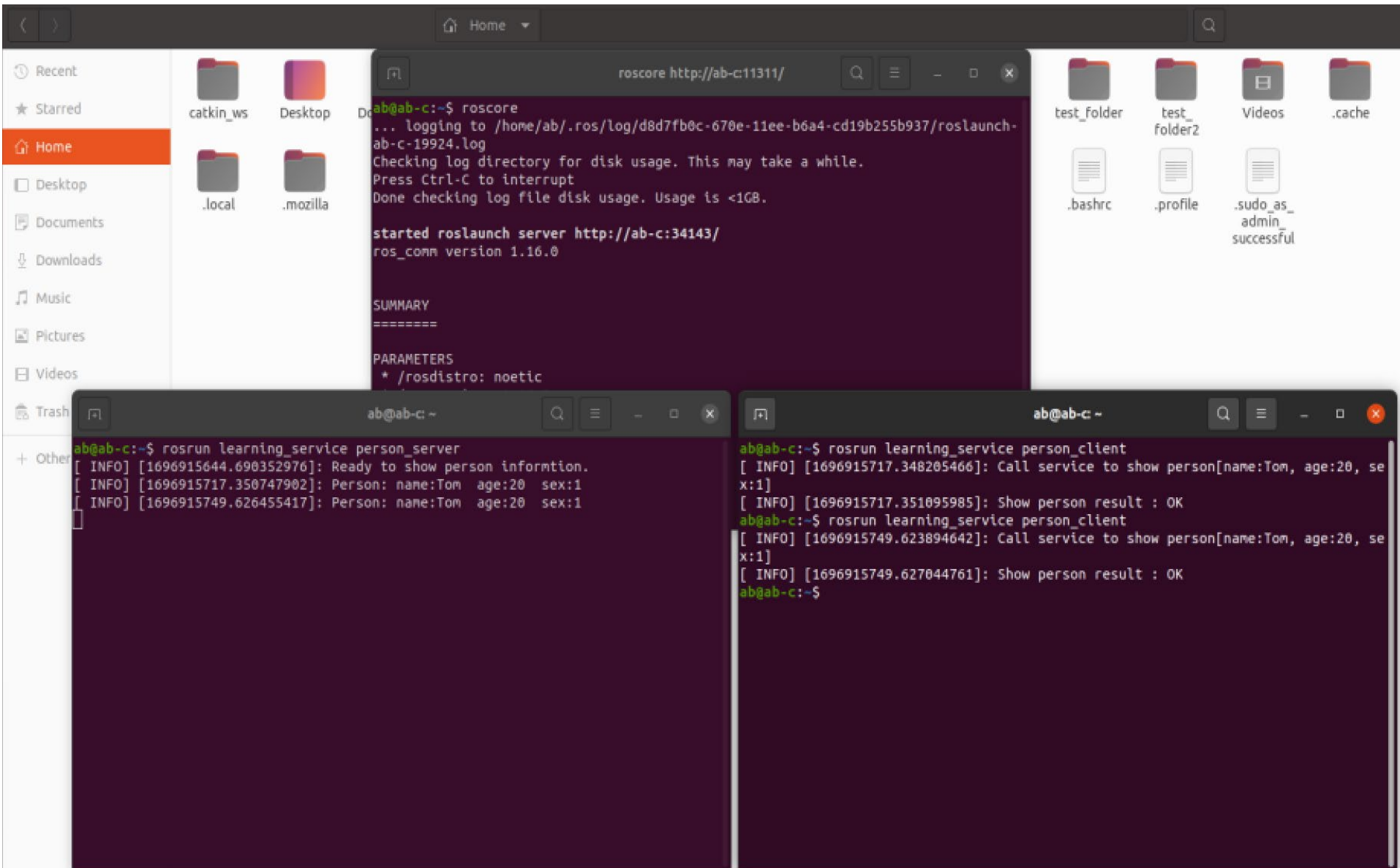
catkin_make Compile

```
ab@ab-c: ~/catkin_ws
ab@ab-c:~/catkin_ws$ catkin_make
Base path: /home/ab/catkin_ws
Source space: /home/ab/catkin_ws/src
Build space: /home/ab/catkin_ws/build
Devel space: /home/ab/catkin_ws/devel
Install space: /home/ab/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/ab/catkin_ws/build"
####
####
#### Running command: "make -j2 -l2" in "/home/ab/catkin_ws/build"
####
[ 0%] Built target std_msgs_generate_messages_cpp
[ 0%] Built target _learning_topic_generate_messages_check_deps_Person
[ 15%] Built target velocity_publisher
[ 15%] Built target std_msgs_generate_messages_eus
[ 15%] Built target std_msgs_generate_messages_py
[ 15%] Built target std_msgs_generate_messages_lisp
[ 15%] Built target std_msgs_generate_messages_nodejs
[ 23%] Built target learning_topic_generate_messages_cpp
[ 38%] Built target learning_topic_generate_messages_eus
ab@ab-c:~$
```


Server and Client Compile

```
$ cd ~/catkin_ws  
$ catkin_make  
$ source devel/setup.bash  
$ roscore  
$ rosrn learning_service person_server  
$ rosrn learning_service person_client
```

Server and Client Compile



The screenshot displays a Linux desktop environment with a file manager in the background and three terminal windows in the foreground. The file manager shows a sidebar with 'Recent', 'Starred', 'Home', 'Desktop', 'Documents', 'Downloads', 'Music', 'Pictures', 'Videos', 'Trash', and 'Other'. The main pane shows a 'Home' directory with folders like 'catkin_ws', 'Desktop', '.local', and '.mozilla', and files like 'test_folder', 'test_folder2', 'Videos', '.cache', '.bashrc', '.profile', and '.sudo_as_admin_successful'.

The top terminal window, titled 'roscore http://ab-c:11311/', shows the following output:

```
ab@ab-c:~$ roscore
... logging to /home/ab/.ros/log/d8d7fb0c-670e-11ee-b6a4-cd19b255b937/roslaunch-ab-c-19924.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ab-c:34143/
ros_comm version 1.16.0

SUMMARY
=====

PARAMETERS
* /rostdistro: noetic
```

The bottom-left terminal window, titled 'ab@ab-c ~', shows the following output:

```
ab@ab-c:~$ roslaunch learning_service person_server
[ INFO] [1696915644.690352976]: Ready to show person informtion.
[ INFO] [1696915717.350747902]: Person: name:Tom age:20 sex:1
[ INFO] [1696915749.626455417]: Person: name:Tom age:20 sex:1
```

The bottom-right terminal window, titled 'ab@ab-c ~', shows the following output:

```
ab@ab-c:~$ roslaunch learning_service person_client
[ INFO] [1696915717.348205466]: Call service to show person[name:Tom, age:20, sex:1]
[ INFO] [1696915717.351095985]: Show person result : OK
ab@ab-c:~$ roslaunch learning_service person_client
[ INFO] [1696915749.623894642]: Call service to show person[name:Tom, age:20, sex:1]
[ INFO] [1696915749.627044761]: Show person result : OK
ab@ab-c:~$
```

Create Client Code Python

```
person_client.py
~/catkin_ws/src/learning_service/scripts

Open [v] [⌕]

1 import sys
2 import rospy
3 from learning_service.srv import Person, PersonRequest
4
5 def person_client():
6
7     rospy.init_node('person_client')
8
9
10    rospy.wait_for_service('/show_person')
11    try:
12        person_client = rospy.ServiceProxy('/show_person', Person)
13
14
15        response = person_client("Tom", 20, PersonRequest.male)
16        return response.result
17    except rospy.ServiceException, e:
18        print "Service call failed: %s"%e
19
20 if __name__ == "__main__":
21
22     print "Show person result : %s" %(person_client())
23
```

Create Server Code Python

```
import rospy

from learning_service.srv import Person, PersonResponse

def personCallback(req):

    rospy.loginfo("Person: name:%s age:%d sex:%d", req.name, req.age, req.sex)

    return PersonResponse("OK")

def person_server():

    rospy.init_node('person_server')

    s = rospy.Service('/show_person', Person, personCallback)

    print "Ready to show person informtion."

    rospy.spin()

if __name__ == "__main__":

    person_server()
```