The Robot Operating System

Day 2 Tutorial II – Programming ROS nodes in C++

Dr. Murilo Fernandes Martins

Department of Electrical Engineering
Centro Universitário da FEI

28 January 2014

Outline

Contents of a catkin package

Recalling the requirements of a catking package

- must contain a catkin compliant package.xml file
- must contain a CMakeLists.txt which uses catkin
- there can be no more than one package in each folder

catkin packages are recommended to reside in a catkin workspace

```
catkin_ws/                  -- WORKSPACE
  src/                      -- SOURCE SPACE
    CMakeLists.txt          -- 'Toplevel' CMake file provide 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
```

Creating a catkin package

Change to the source space directory of the catkin workspace

```
$ cd ~/catkin_ws/src
```

Creating a catkin package

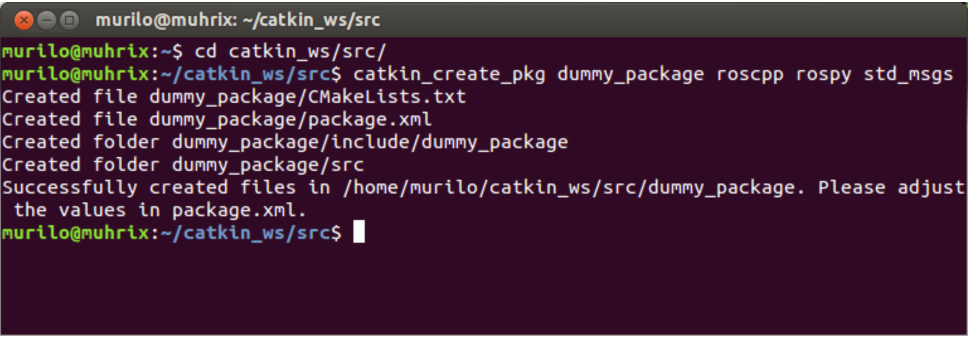
- catkin_create_pkg is a script which creates the bare minimum requirements of a catkin package

```
$ catkin_create_pkg <package_name> [depend1] [depend2] ...
```

- create a dummy catkin package with some dependencies

```
$ catkin_create_pkg dummy_package roscpp rospy std_msgs
```

The output should look like this. . .


murilo@muhrix: ~/catkin_ws/src

```
murilo@muhrix:~$ cd catkin_ws/src/
murilo@muhrix:~/catkin_ws/src$ catkin_create_pkg dummy_package roscpp rospy std_msgs
Created file dummy_package/CMakeLists.txt
Created file dummy_package/package.xml
Created folder dummy_package/include/dummy_package
Created folder dummy_package/src
Successfully created files in /home/murilo/catkin_ws/src/dummy_package. Please adjust
 the values in package.xml.
murilo@muhrix:~/catkin_ws/src$ 
```

Package dependencies

First-order dependencies

- the **first-order** dependencies can be reviewed with rospack

```
$ rospack depends1 dummy_package
```

package.xml

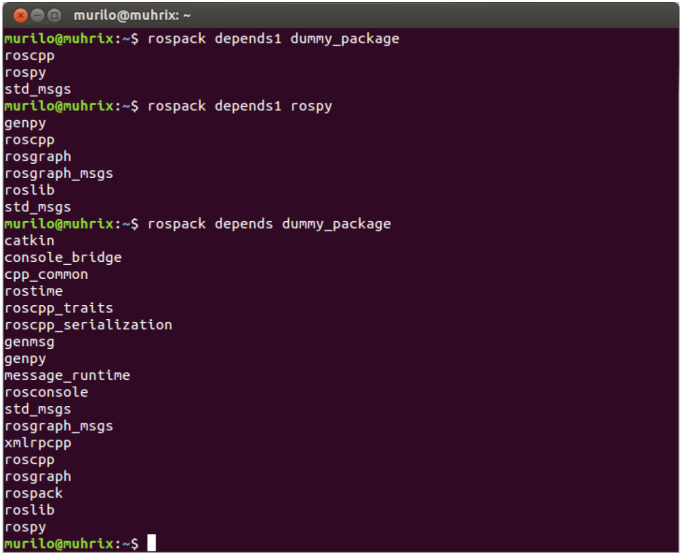
- The **first-order** dependencies are stored in package.xml

Indirect dependencies

- in many cases, dependencies also have their own dependencies
- for instance, the **rospy** package has other dependencies
- rospack can also show indirect dependencies of a package

```
$ rospack depends1 rospy
$ rospack depends dummy_package
```

The output should look like this. . .



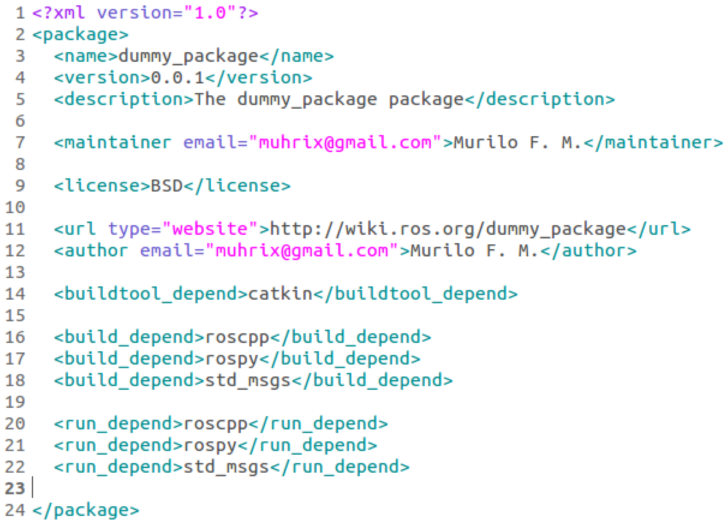
```
murilo@muhrix:~$ rospack depends1 dummy_package
roscpp
rospy
std_msgs
murilo@muhrix:~$ rospack depends1 rospy
genpy
roscpp
rosgraph
rosgraph_msgs
roslib
std_msgs
murilo@muhrix:~$ rospack depends dummy_package
catkin
console_bridge
cpp_common
rostime
roscpp_traits
roscpp_serialization
genmsg
genpy
message_runtime
rosconsole
std_msgs
rosgraph_msgs
xmlrpcpp
roscpp
rosgraph
rospack
roslib
rospy
murilo@muhrix:~$ 
```

Configuring the catkin package

Customising the package.xml file

- description tag: required (by convention, short sentence)
- maintainer tags: required and important (know who to contact)
- license tags: required (e.g., BSD, GPLv2, LGPLv3, . . .)
- url tags: optional (e.g., repository, issue tracker, website)
- author tags: optional
- dependencies tags: required (buildtool, build, run and test)

Final package.xml for dummy_package



```xml
<?xml version="1.0"?>
<package>
  <name>dummy_package</name>
  <version>0.0.1</version>
  <description>The dummy_package package</description>

  <maintainer email="muhrix@gmail.com">Murilo F. M.</maintainer>

  <license>BSD</license>

  <url type="website">http://wiki.ros.org/dummy_package</url>
  <author email="muhrix@gmail.com">Murilo F. M.</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>

</package>
```

Customising the CMakeLists.txt

A note on dependencies

- ROS was installed with apt-get; dependencies are already installed
- there might be cases which not all dependencies are installed
- in which case, rosdep can be used to install package dependencies
- keep in mind that rosdep can install debian packages only

Customising the CMakeLists.txt

A note on dependencies

- ROS was installed with apt-get; dependencies are already installed
- there might be cases which not all dependencies are installed
- in which case, rosdep can be used to install package dependencies
- keep in mind that rosdep can install debian packages only

CMakeLists.txt

- input to the CMake build system for building software packages
- describe how to build the code and where to install binaries
- in catkin, it is a standard CMakeLists.txt with additional constraints

Customising the CMakeLists.txt

Overall structure and ordering of a CMakeLists.txt

1. required CMake version: cmake_minimum_required
2. package name: project()
3. find other CMake/catkin packages required for build: find_package()
4. message/service/action generators:
   add_message_files(), add_service_files(), add_action_files()
5. invoke message/service/action generation: generate_messages()
6. specify package build info export: catkin_package()
7. libraries/executables to build:
   add_library(), add_executable(), target_link_libraries()
8. tests to build: catkin_add_gtest()
9. install rules: install()

Customising the CMakeLists.txt

CMake version

- catkin requires version 2.8.3 or higher

Customising the CMakeLists.txt

CMake version

- catkin requires version 2.8.3 or higher

Package name

- name of the package
    - must match name in package.xml
    - and also name of package directory

Customising CMakeLists.txt

Dependent CMake package

- find_package() specifies other CMake packages needed to build package
- there is always, at least, one dependency on catkin:

```
find_package(catkin REQUIRED)
```

- other wet packages are also catkin components, e.g.:

```
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)
```

- find_package() should be used with components which build flags are needed; runtime dependencies should not be added
- Boost, OpenCV and PCL are not catkin components

```
find_package(Boost REQUIRED COMPONENTS signals thread)
```

Customising CMakeLists.txt

catkin_package()

- catkin-provided CMake macro
- required to specify catkin-specific info to the build system
- which in turn is used to generate pkg-config and CMake files
- must be called before declaring targets
- optional arguments are:
  - INCLUDE_DIRS: exported include paths
  - LIBRARIES: exported libraries
  - CATKIN_DEPENDS: other catkin packages (dependencies)
  - DEPENDS: non-catkin CMake (system) dependencies
  - CFG_EXTRAS: additional configuration options

Customising CMakeLists.txt

Build targets

- include paths

```
include_directories(include ${Boost_INCLUDE_DIRS}
                            ${catkin_INCLUDE_DIRS})
```

- (shared) library targets

```
add_library(dummy src/dummy.cpp)
```

- executable targets

```
add_executable(dummy_node src/main.cpp src/some_functions.cpp)
```

- target_link_ilbraries

```
target_link_libraries(dummy ${catkin_LIBRARIES})
target_link_libraries(dummy_node dummy ${Boost_LIBRARIES})
```

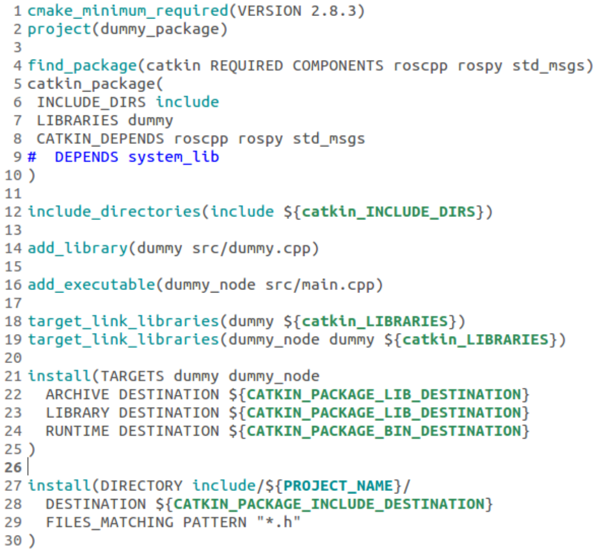
Customising CMakeLists.txt

Suggested homework

- we have not talked about:
  - messages, services and action targets
  - unit tests
  - installable targets

For more information, see CMakeLists.txt ROS wiki page

Hypothetical CMakeLists.txt for dummy_package



```
cmake_minimum_required(VERSION 2.8.3)
project(dummy_package)

find_package(catkin REQUIRED COMPONENTS roscpp rospy std_msgs)
catkin_package(
 INCLUDE_DIRS include
 LIBRARIES dummy
 CATKIN_DEPENDS roscpp rospy std_msgs
#  DEPENDS system_lib
)

include_directories(include ${catkin_INCLUDE_DIRS})

add_library(dummy src/dummy.cpp)

add_executable(dummy_node src/main.cpp)

target_link_libraries(dummy ${catkin_LIBRARIES})
target_link_libraries(dummy_node dummy ${catkin_LIBRARIES})

install(TARGETS dummy dummy_node
  ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)

install(DIRECTORY include/${PROJECT_NAME}/
  DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
  FILES_MATCHING PATTERN "*.h"
)
```

Building the catkin workspace

Simple steps to build the catkin workspace

```
$ cd ~/catkin_ws
$ catkin_make
```

Analysing changes after building the workspace

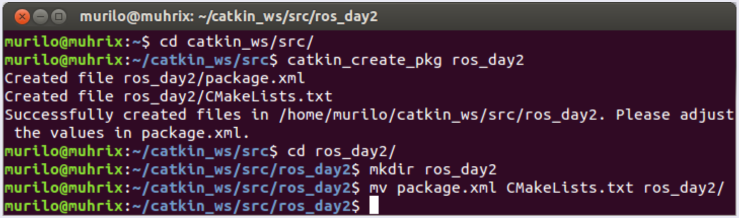
Two new directories were created:

- build/: where cmake and make are called to configure and build the packages within ~/catkin_ws/src
- devel/: where executables, libraries and header files go prior to installation

Eclipse IDE – preliminaries

Create a metapackage

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg ros_day2
$ cd ros_day2
$ mv package.xml CMakeLists.txt ros_day2/
```



```
murilo@muhrix: ~/catkin_ws/src/ros_day2
murilo@muhrix:~$ cd catkin_ws/src/
murilo@muhrix:~/catkin_ws/src$ catkin_create_pkg ros_day2
Created file ros_day2/package.xml
Created file ros_day2/CMakeLists.txt
Successfully created files in /home/murilo/catkin_ws/src/ros_day2. Please adjust
 the values in package.xml.
murilo@muhrix:~/catkin_ws/src$ cd ros_day2/
murilo@muhrix:~/catkin_ws/src/ros_day2$ mkdir ros_day2
murilo@muhrix:~/catkin_ws/src/ros_day2$ mv package.xml CMakeLists.txt ros_day2/
murilo@muhrix:~/catkin_ws/src/ros_day2$ 
```

The metapackage will contain two packages

- day2_talker
- day2_listener

Eclipse IDE – preliminaries

Modify the the contents of package.xml

```xml
<?xml version="1.0"?>
<package>
  <name>ros_day2</name>
  <version>0.0.1</version>
  <description>The ros_day2 metapackage</description>
  <maintainer email="muhrix@gmail.com">Murilo F. M.</maintainer>
  <license>BSD</license>
  <author email="muhrix@gmail.com">Murilo F. M.</author>
  <buildtool_depend>catkin</buildtool_depend>
  <run_depend>day2_talker</run_depend>
  <run_depend>day2_listener</run_depend>
  <export>
    <metapackage/>
  </export>
</package>
```

Modify the the contents of CMakeLists.txt

```cmake
cmake_minimum_required(VERSION 2.8.3)
project(ros_day2)
find_package(catkin REQUIRED)
catkin_metapackage()
```

Eclipse IDE – preliminaries

Create packages within ros_day2 metapackage

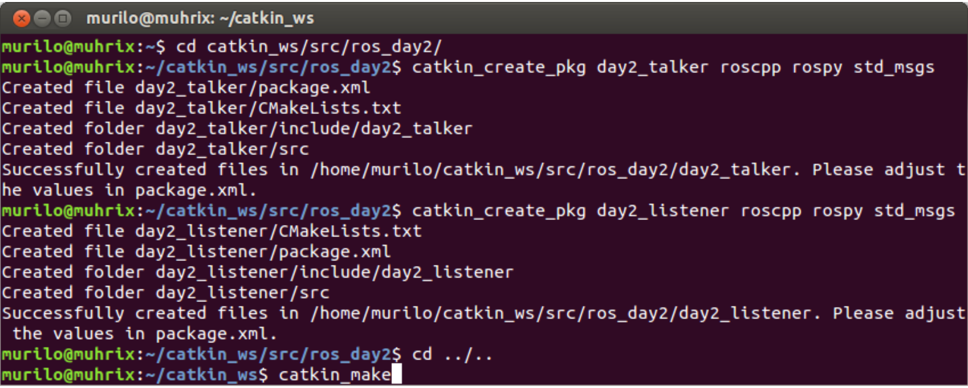
```
$ cd ~/catkin_ws/src/ros_day2
$ catkin_create_pkg day2_talker roscpp rospy std_msgs
$ catkin_create_pkg day2_listener roscpp rospy std_msgs
```

Build the catkin workspace

```
$ cd ~/catkin_ws
$ catkin_make
```

The output should look like this. . .

The workspace should build with no errors at this stage



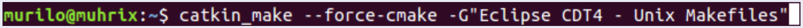
murilo@muhrix: ~/catkin_ws

```
murilo@muhrix:~$ cd catkin_ws/src/ros_day2/
murilo@muhrix:~/catkin_ws/src/ros_day2$ catkin_create_pkg day2_talker roscpp rospy std_msgs
Created file day2_talker/package.xml
Created file day2_talker/CMakeLists.txt
Created folder day2_talker/include/day2_talker
Created folder day2_talker/src
Successfully created files in /home/murilo/catkin_ws/src/ros_day2/day2_talker. Please adjust t
he values in package.xml.
murilo@muhrix:~/catkin_ws/src/ros_day2$ catkin_create_pkg day2_listener roscpp rospy std_msgs
Created file day2_listener/CMakeLists.txt
Created file day2_listener/package.xml
Created folder day2_listener/include/day2_listener
Created folder day2_listener/src
Successfully created files in /home/murilo/catkin_ws/src/ros_day2/day2_listener. Please adjust
 the values in package.xml.
murilo@muhrix:~/catkin_ws/src/ros_day2$ cd ../..
murilo@muhrix:~/catkin_ws$ catkin_make
```

Generate Eclipse project files

Eclipse project files for C++ (whole catkin workspace)

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



```
murilo@muhrix:~$ catkin_make --force-cmake -G"Eclipse CDT4 - Unix Makefiles"
```

Eclipse project file for Python (per ROS package)

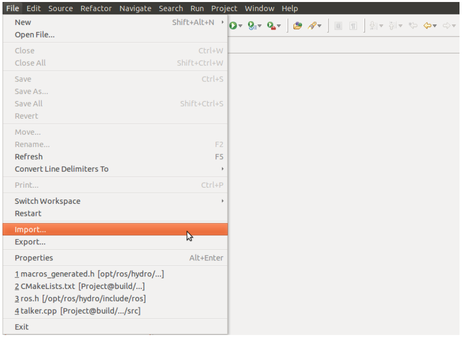
```
$ cd ~/catkin_ws/src/ros_day2/day2_talker
$ python $(rospack find mk)/make_pydev_project.py
$ cd ../day2_listener
$ python $(rospack find mk)/make_pydev_project.py
```

Load Eclipse IDE

- press “windows” key and type “Eclipse”

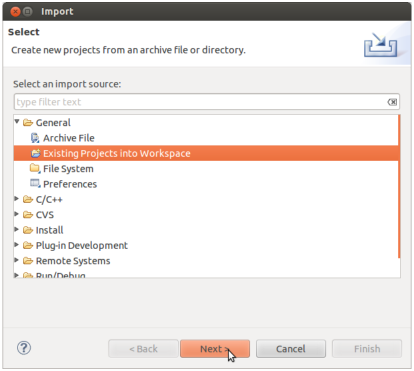
Import catkin workspace into Eclipse

1. Go to menu **File** → **Import...**


File Edit Source Refactor Navigate Search Run Project Window Help
New Shift+Alt+N
Open File...
Close Ctrl+W
Close All Shift+Ctrl+W
Save Ctrl+S
Save As...
Save All Shift+Ctrl+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace
Restart
Import...
Export...
Properties Alt+Enter
1 macros_generated.h [opt/ros/hydro/...]
2 CMakeLists.txt [Project@build/...]
3 ros.h [/opt/ros/hydro/include/ros]
4 talker.cpp [Project@build/.../src]
Exit

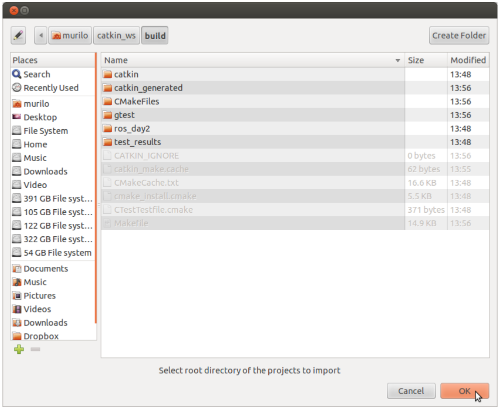
Import catkin workspace into Eclipse

1. Go to menu **File** → **Import...**
2. Expand **General**, then select **Existing Projects into Workspace**


Import
Select
Create new projects from an archive file or directory.
Select an import source:
type filter text

Import catkin workspace into Eclipse

1. Go to menu **File** → **Import...**
2. Expand **General**, then select **Existing Projects into Workspace**
3. Navigate to: ~/catkin_ws/build/ and select that directory

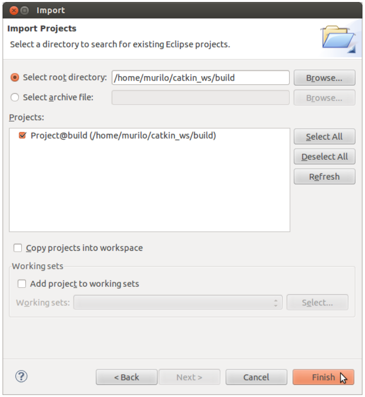


Name	Size	Modified
catkin		13:48
catkin_generated		13:56
CMakeFiles		13:56
gtest		13:56
ros_day2		13:48
test_results		13:48
CATKIN_IGNORE	0 bytes	13:56
catkin_make.cache	62 bytes	13:55
CMakeCache.txt	16.6 KB	13:48
cmake_install.cmake	5.5 KB	13:48
CTestTestfile.cmake	371 bytes	13:48
Makefile	14.9 KB	13:56

Select root directory of the projects to import

Import catkin workspace into Eclipse

1. Go to menu **File** → **Import...**
2. Expand **General**, then select **Existing Projects into Workspace**
3. Navigate to: ∼/catkin_ws/build/ and select that directory
4. Click **Finish**


Import
Import Projects
Select a directory to search for existing Eclipse projects.
[✓ Select root directory:] /home/murilo/catkin_ws/build
Select archive file:
Projects:
✓ Project@build (/home/murilo/catkin_ws/build)
☐ Copy projects into workspace
Working sets
☐ Add project to working sets
Working sets:

Do not select **Copy projects into Workspace**

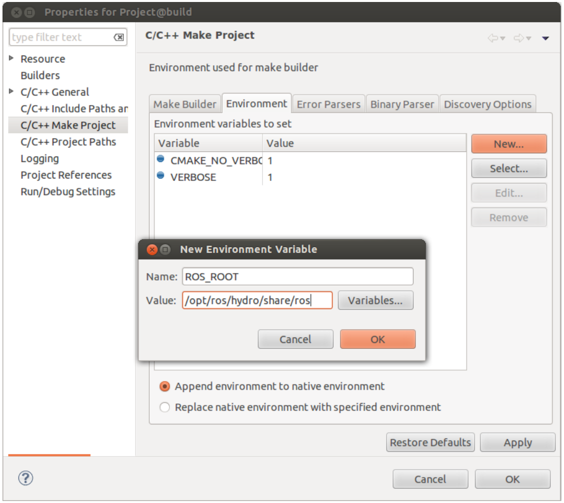
Build catkin workspace inside Eclipse

Configure environment variables

- right click on the project and select **Properties**
- select **C/C++ Make Project** → **Environment** tab
- Click **New...** and add the following environment variables:
  - ROS_ROOT
  - ROS_PACKAGE_PATH
  - PYTHONPATH
  - PATH
- the values for such variables can be easily obtained:

```
$ echo $ROS_ROOT
$ echo $ROS_PACKAGE_PATH
$ echo $PYTHONPATH
$ echo $PATH
```

Build catkin workspace inside Eclipse



Properties for Project@build

C/C++ Make Project

Environment used for make builder

Environment variables to set

Variable	Value
CMAKE_NO_VERBC	1
VERBOSE	1

New Environment Variable

Name: ROS_ROOT

Value: /opt/ros/hydro/share/ros

[✓ Append environment to native environment]

Replace native environment with specified environment

Press **CTRL-B** (or select **Project** → **Build project** in the menu)

Check list

Checking what has been done so far and what comes next

✓ create catkin workspace

Check list

Checking what has been done so far and what comes next

- ✓ create catkin workspace
- ✓ source devel/setup.bash (from within ~/catkin_ws)

Check list

Checking what has been done so far and what comes next

- ✓ create catkin workspace
- ✓ source devel/setup.bash (from within ~/catkin_ws)
- ✓ configure Eclipse IDE

Check list

Checking what has been done so far and what comes next

- ✓ create catkin workspace
- ✓ source devel/setup.bash (from within ~/catkin_ws)
- ✓ configure Eclipse IDE
- ✓ create metapackage ros_day2

Check list

Checking what has been done so far and what comes next

- ✓ create catkin workspace
- ✓ source devel/setup.bash (from within ~/catkin_ws)
- ✓ configure Eclipse IDE
- ✓ create metapackage ros_day2
  - ✓ modify package.xml
  - ✓ modify CMakeLists.txt

Check list

Checking what has been done so far and what comes next

- ✓ create catkin workspace
- ✓ source devel/setup.bash (from within ∼/catkin_ws)
- ✓ configure Eclipse IDE
- ✓ create metapackage ros_day2
  - ✓ modify package.xml
  - ✓ modify CMakeLists.txt
- ✓ create package day2_talker within ros_day2

Check list

Checking what has been done so far and what comes next

- ✓ create catkin workspace
- ✓ source devel/setup.bash (from within ~/catkin_ws)
- ✓ configure Eclipse IDE
- ✓ create metapackage ros_day2
  - ✓ modify package.xml
  - ✓ modify CMakeLists.txt
- ✓ create package day2_talker within ros_day2
  - × modify package.xml
  - × modify CMakeLists.txt
  - × add source files to package (e.g., talker_node.cpp)

Check list

Checking what has been done so far and what comes next

- ✓ create catkin workspace
- ✓ source devel/setup.bash (from within ~/catkin_ws)
- ✓ configure Eclipse IDE
- ✓ create metapackage ros_day2
  - ✓ modify package.xml
  - ✓ modify CMakeLists.txt
- ✓ create package day2_talker within ros_day2
  - × modify package.xml
  - × modify CMakeLists.txt
  - × add source files to package (e.g., talker_node.cpp)
- ✓ create package day2_listener within ros_day2
  - × modify package.xml
  - × modify CMakeLists.txt
  - × add source files to package (e.g., listener_node.cpp)

Check list

Modify package.xml and CMakeLists.txt

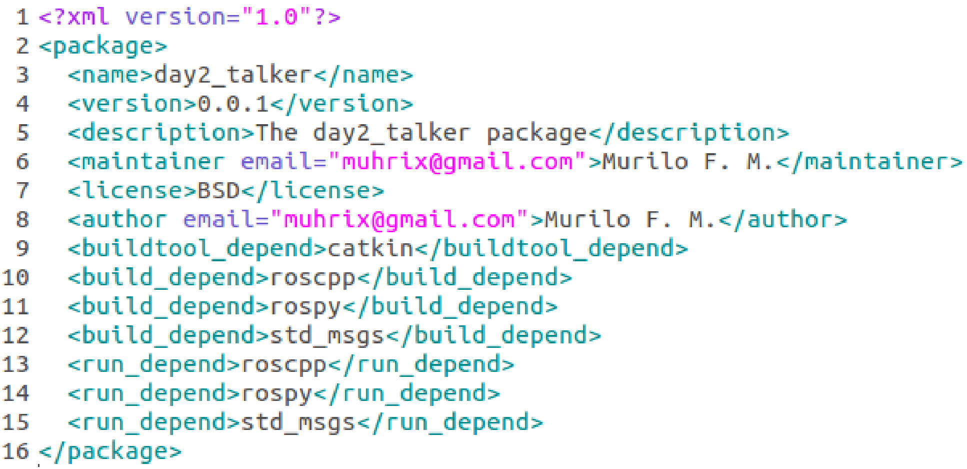
- this was covered in the **dummy_package** example
- make the changes considering:
  - talker_node.cpp will be created in day2_talker/src/
  - listener_node.cpp will be created in day2_listener/src/

Add source files to packages (Eclipse)

- expand **Project@Build** → **[Source directory]** → **ros_day2** → **day2_talker**/**day2_listener**
- right click **src** → **new. . .** → **Source File**
- name it talker_node.cpp/listener_node.cpp

Alternatively, the files can be created using any text editor.

day2_talker package.xml

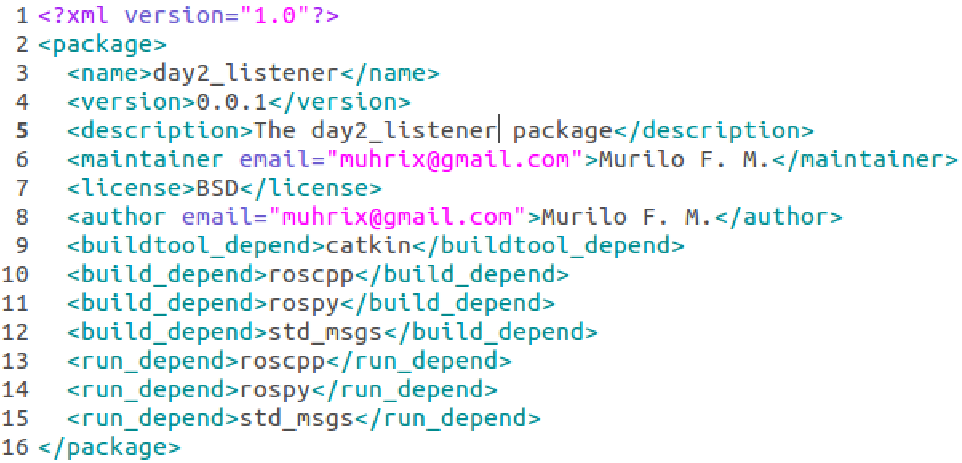


```xml
<?xml version="1.0"?>
<package>
  <name>day2_talker</name>
  <version>0.0.1</version>
  <description>The day2_talker package</description>
  <maintainer email="muhrix@gmail.com">Murilo F. M.</maintainer>
  <license>BSD</license>
  <author email="muhrix@gmail.com">Murilo F. M.</author>
  <buildtool_depend>catkin</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>
</package>
```

day2_talker CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(day2_talker)

find_package(catkin REQUIRED COMPONENTS roscpp rospy std_msgs)

catkin_package(
 CATKIN_DEPENDS roscpp rospy std_msgs
)

include_directories(${catkin_INCLUDE_DIRS})
add_executable(talker_node src/talker_node.cpp)
target_link_libraries(talker_node ${catkin_LIBRARIES})

install(TARGETS talker_node
  ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

day2_listener package.xml

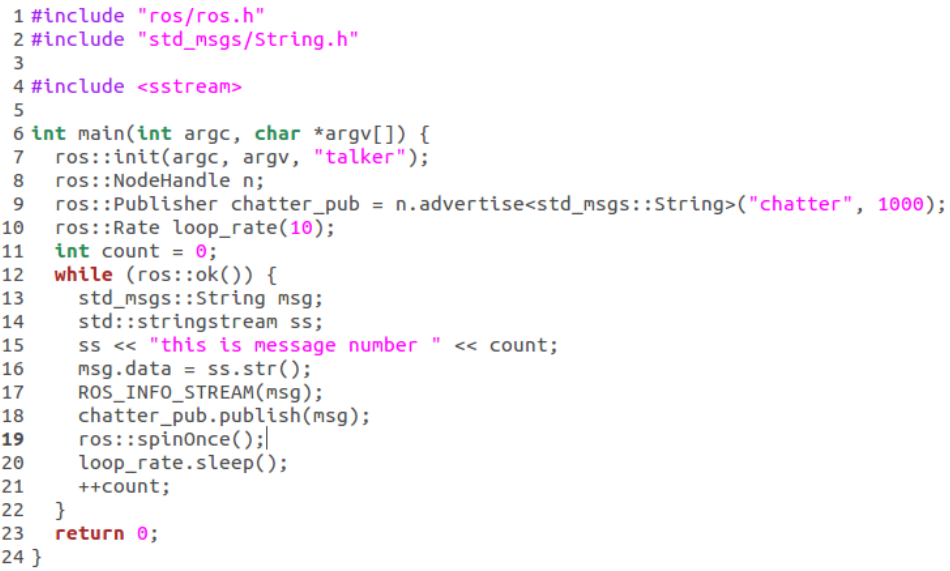


```xml
<?xml version="1.0"?>
<package>
  <name>day2_listener</name>
  <version>0.0.1</version>
  <description>The day2_listener package</description>
  <maintainer email="muhrix@gmail.com">Murilo F. M.</maintainer>
  <license>BSD</license>
  <author email="muhrix@gmail.com">Murilo F. M.</author>
  <buildtool_depend>catkin</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>
</package>
```

day2_listener CMakeLists.txt

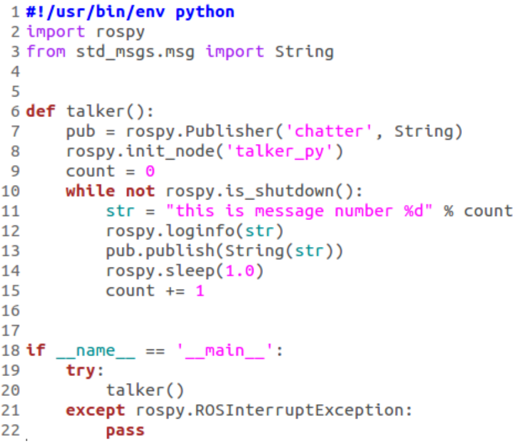
```
cmake_minimum_required(VERSION 2.8.3)
project(day2_listener)

find_package(catkin REQUIRED COMPONENTS roscpp rospy std_msgs)

catkin_package(
 CATKIN_DEPENDS roscpp rospy std_msgs
)

include_directories(${catkin_INCLUDE_DIRS})
add_executable(listener_node src/listener_node.cpp)
target_link_libraries(listener_node ${catkin_LIBRARIES})

install(TARGETS listener_node
  ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

Writing a simple publisher in C++



```cpp
#include "ros/ros.h"
#include "std_msgs/String.h"

#include <sstream>

int main(int argc, char *argv[]) {
  ros::init(argc, argv, "talker");
  ros::NodeHandle n;
  ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);
  ros::Rate loop_rate(10);
  int count = 0;
  while (ros::ok()) {
    std_msgs::String msg;
    std::stringstream ss;
    ss << "this is message number " << count;
    msg.data = ss.str();
    ROS_INFO_STREAM(msg);
    chatter_pub.publish(msg);
    ros::spinOnce();
    loop_rate.sleep();
    ++count;
  }
  return 0;
}
```

Writing a simple publisher in Python

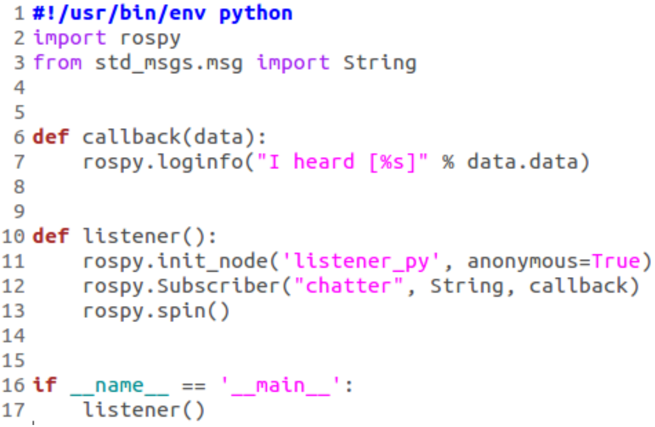


```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String


def talker():
    pub = rospy.Publisher('chatter', String)
    rospy.init_node('talker_py')
    count = 0
    while not rospy.is_shutdown():
        str = "this is message number %d" % count
        rospy.loginfo(str)
        pub.publish(String(str))
        rospy.sleep(1.0)
        count += 1


if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Writing a simple subscriber in C++

```cpp
#include "ros/ros.h"
#include "std_msgs/String.h"

void chatterCallback(const std_msgs::String::ConstPtr& msg) {
  ROS_INFO_STREAM("I heard: [" << msg->data << "]");
}

int main(int argc, char *argv[]) {
  ros::init(argc, argv, "listener");
  ros::NodeHandle n;

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

  ros::spin();

  return 0;
}
```

Writing a simple subscriber in Python



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


def callback(data):
    rospy.loginfo("I heard [%s]" % data.data)


def listener():
    rospy.init_node('listener_py', anonymous=True)
    rospy.Subscriber("chatter", String, callback)
    rospy.spin()


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

Testing the publisher/subscriber

Building the workspace, sourcing the shell and running ROS Master

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
```

Running the publisher (in a new Terminal)

```
$ rosrun day2_talker talker_node
```

Running the subscriber (in a new Terminal)

```
$ rosrun day2_listener listener_node
```

Testing the publisher/subscriber

Building the workspace, sourcing the shell and running ROS Master

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
```

Running the publisher (in a new Terminal)

```
$ rosrun day2_talker talker_node
```

Running the subscriber (in a new Terminal)

```
$ rosrun day2_listener listener_node
```

Can we add another callback and subscribe to the same topic?

Before proceeding to the service/client example

Create two new packages within the ros_day2 metapackage

```
$ cd ~/catkin_ws/src/ros_day2
$ catkin_create_pkg day2_client roscpp rospy std_msgs
$ catkin_create_pkg day2_service roscpp rospy std_msgs
$ cd day2_service/
$ mkdir srv
```

Create the AddTwoInts.srv service (similar process for messages)

- the contents of this file should be:

```
int64 a
int64 b
---
int64 sum
```

Before proceeding to the service/client example

Modify package.xml

- day2_service must have:
  - build dependency on **message_generation**
  - run dependency on **message_runtime**
- day2_client must have:
  - build and run dependencies on **day2_service**
- ros_day2 metapackage must have run dependencies on:
  - **day2_service**
  - **day2_client**

Before proceeding to the service/client example

Modify CMakeLists.txt

- make the changes considering:
  - service_node.cpp will be created in day2_service/src/
  - client_node.cpp will be created in day2_client/src/
- message_generation is a REQUIRED catkin COMPONENT
- CATKIN_DEPENDS must export message_runtime
- include_directories() must have include
- for day2_service, remember the required macros:
  - add_service_files()
  - generate_messages()
  - add_dependencies()

Remember to add the source files to the src/ folder of the packages

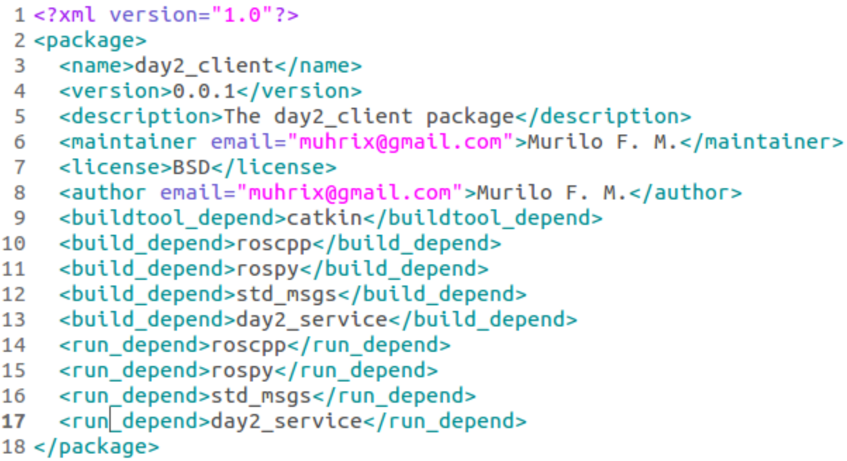
day2_service package.xml

```xml
<?xml version="1.0"?>
<package>
  <name>day2_service</name>
  <version>0.0.1</version>
  <description>The day2_service package</description>
  <maintainer email="muhrix@gmail.com">Murilo F. M.</maintainer>
  <license>BSD</license>
  <author email="muhrix@gmail.com">Murilo F. M.</author>
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>message_generation</build_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>message_runtime</run_depend>
</package>
```

day2_service CMakeLists.txt

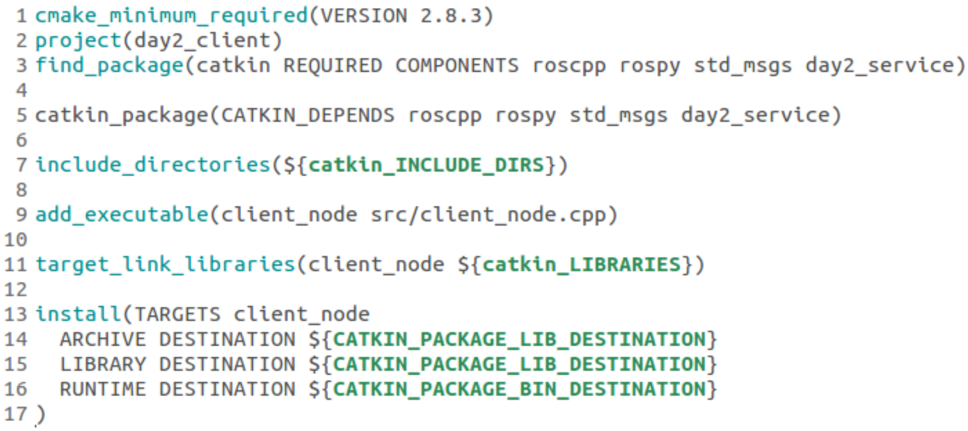
```
cmake_minimum_required(VERSION 2.8.3)
project(day2_service)

find_package(catkin REQUIRED COMPONENTS roscpp rospy std_msgs message_generation)

add_service_files(
  FILES
  AddTwoInts.srv
)
generate_messages( DEPENDENCIES std_msgs)
catkin_package(CATKIN_DEPENDS roscpp rospy std_msgs message_runtime)

include_directories(include ${catkin_INCLUDE_DIRS})
add_executable(service_node src/service_node.cpp)
add_dependencies(service_node day2_service_generate_messages_cpp)
target_link_libraries(service_node ${catkin_LIBRARIES})

install(TARGETS service_node
  ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)

install(DIRECTORY include/${PROJECT_NAME}/
  DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
  FILES_MATCHING PATTERN "*.h"
)
```

day2_client package.xml



```xml
<?xml version="1.0"?>
<package>
  <name>day2_client</name>
  <version>0.0.1</version>
  <description>The day2_client package</description>
  <maintainer email="muhrix@gmail.com">Murilo F. M.</maintainer>
  <license>BSD</license>
  <author email="muhrix@gmail.com">Murilo F. M.</author>
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>day2_service</build_depend>
  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>
  <run_depend>day2_service</run_depend>
</package>
```

day2_client CMakeLists.txt

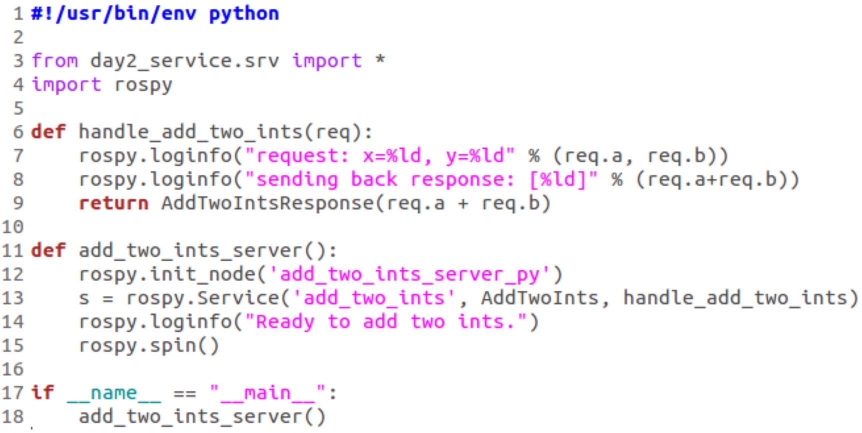


```
cmake_minimum_required(VERSION 2.8.3)
project(day2_client)
find_package(catkin REQUIRED COMPONENTS roscpp rospy std_msgs day2_service)

catkin_package(CATKIN_DEPENDS roscpp rospy std_msgs day2_service)

include_directories(${catkin_INCLUDE_DIRS})

add_executable(client_node src/client_node.cpp)

target_link_libraries(client_node ${catkin_LIBRARIES})

install(TARGETS client_node
  ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
  RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

Writing a service node in C++

```
#include "ros/ros.h"
#include "day2_service/AddTwoInts.h"

bool add(day2_service::AddTwoInts::Request  &req,
         day2_service::AddTwoInts::Response &res) {
  res.sum = req.a + req.b;
  ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
  ROS_INFO("sending back response: [%ld]", (long int)res.sum);
  return true;
}

int main(int argc, char *argv[]) {
  ros::init(argc, argv, "add_two_ints_server");
  ros::NodeHandle n;

  ros::ServiceServer service = n.advertiseService("add_two_ints", add);
  ROS_INFO("Ready to add two ints.");
  ros::spin();

  return 0;
}
```

Writing a service node in Python



```
#!/usr/bin/env python

from day2_service.srv import *
import rospy

def handle_add_two_ints(req):
    rospy.loginfo("request: x=%ld, y=%ld" % (req.a, req.b))
    rospy.loginfo("sending back response: [%ld]" % (req.a+req.b))
    return AddTwoIntsResponse(req.a + req.b)

def add_two_ints_server():
    rospy.init_node('add_two_ints_server_py')
    s = rospy.Service('add_two_ints', AddTwoInts, handle_add_two_ints)
    rospy.loginfo("Ready to add two ints.")
    rospy.spin()

if __name__ == "__main__":
    add_two_ints_server()
```

Writing a client node in C++

```cpp
#include "ros/ros.h"
#include "day2_service/AddTwoInts.h"
#include <cstdlib>

int main(int argc, char *argv[]) {
  ros::init(argc, argv, "add_two_ints_client");
  if (argc != 3) {
    ROS_INFO("usage: add_two_ints_client X Y");
    return 1;
  }
  ros::NodeHandle n;
  ros::ServiceClient client = n.serviceClient<day2_service::AddTwoInts>("add_two_ints");
  day2_service::AddTwoInts srv;
  srv.request.a = atoll(argv[1]);
  srv.request.b = atoll(argv[2]);
  if (client.call(srv)) {
    ROS_INFO("Sum: %ld", (long int)srv.response.sum);
  }
  else {
    ROS_ERROR("Failed to call service add_two_ints");
    return 1;
  }

  return 0;
}
```

Writing a client node in Python

```python
#!/usr/bin/env python
import roslib
import sys
import rospy
from day2_service.srv import *

def add_two_ints_client(x, y):
    rospy.wait_for_service('add_two_ints')
    try:
        add_two_ints = rospy.ServiceProxy('add_two_ints', AddTwoInts)
        resp1 = add_two_ints(x, y)
        return resp1.sum
    except rospy.ServiceException, e:
        rospy.loginfo("Failed to call service add_two_ints: %s"%e)

def usage():
    return "%s [x y]"%sys.argv[0]

if __name__ == "__main__":
    if len(sys.argv) == 3:
        x = int(sys.argv[1])
        y = int(sys.argv[2])
    else:
        print usage()
        sys.exit(1)
    rospy.loginfo("Sum: %ld"%add_two_ints_client(x,y))
```

Testing the service/client

Building the workspace, sourcing the shell and running ROS Master

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
```

Running the service (in a new Terminal)

```
$ rosrun day2_service service_node
```

Running the client (in a new Terminal)

```
$ rosrun day2_client client_node 2 5
```