cmake\_minimum\_required(VERSION 2.8.3) //版本

project(beginner\_tutorials) //软件包名

## Find catkin macros and libraries //查找catkin宏和库

## if COMPONENTS list like find\_package(catkin REQUIRED COMPONENTS xyz)

## is used, also find other catkin packages

##如果COMPONENTS列表像find\_package一样被使用，也要找到其他的catkin包

find\_package(catkin REQUIRED COMPONENTS

roscpp

rospy

std\_msgs

message\_generation

)

##寻找依赖软件包，在创建包时的依赖项以及后期加入的依赖软件包

## System dependencies are found with CMake's conventions

##使用CMake公约找到系统依赖项

# find\_package(Boost REQUIRED COMPONENTS system)

##find\_package(Boost REQUIRED COMPONENTS 系统)

## Uncomment this if the package has a setup.py. //如果软件包有setup.py，请取消注释

## This macro ensures modules and global scripts declared therein get installed

##这个宏确保在其中声明的模块和全局脚本被安装

## See http://ros.org/doc/api/catkin/html/user\_guide/setup\_dot\_py.html

# catkin\_python\_setup()

################################################

## Declare ROS messages, services and actions ##

##声明ROS消息，服务和操作##

################################################

## To declare and build messages, services or actions from within this package, follow these steps:

##要在此程序包中声明和构建消息，服务或操作，请按照下列步骤操作：

## \* 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, ...).

##让MSG\_DEP\_SET是您在消息/服务/操作中使用的消息类型（例如std\_msgs，actionlib msgs，...）的一组包。

## \* In the file package.xml: //在package.xml中：

## \* add a build\_depend tag for "message\_generation"

## 为message\_generation添加一个build\_depend标记

## \* add a build\_depend and a run\_depend tag for each package in MSG\_DEP\_SET

##为MSG\_DEP\_SET中的每个包添加一个build\_depend和一个run\_depend标记

## \* If MSG\_DEP\_SET isn't empty the following dependency has been pulled in but can be declared for certainty nonetheless:

## 如果MSG\_DEP\_SET不为空，那么下面的依赖关系已被引入，但仍可以声明为确定性：

## \* add a run\_depend tag for "message\_runtime"

## 为message\_runtime 添加一个run\_depend 标签

## \* In this file (CMakeLists.txt): //在这个文件中（CMakeLists.txt）：

## \* add "message\_generation" and every package in MSG\_DEP\_SET to

## find\_package(catkin REQUIRED COMPONENTS ...)

## 添加“message\_generation”和MSG\_DEP\_SET中的每个包到find\_package（catkin REQUIRED COMPONENTS ...）

## \* add "message\_runtime" and every package in MSG\_DEP\_SET to

## catkin\_package(CATKIN\_DEPENDS ...)

## 添加“message\_runtime”和MSG\_DEP\_SET中的每个包catkin\_package（CATKIN DEPENDS ...）

## \* uncomment the add\_\*\_files sections below as needed

## and list every .msg/.srv/.action file to be processed

## 根据需要取消注释以下add\_files部分，并列出要处理的每个.msg / .srv / .action文件

## \* uncomment the generate\_messages entry below

## 取消注释下面的generate\_messages条目

## \* add every package in MSG\_DEP\_SET to generate\_messages(DEPENDENCIES ...)

## 添加MSG\_DEP\_SET中的每个包到generate\_messages（DEPENDENCIES ...）

## Generate messages in the 'msg' folder //在“msg”文件夹中生成消息

add\_message\_files(

FILES

Num.msg

)

## Generate services in the 'srv' folder //在“srv”文件夹中生成服务

add\_service\_files(

FILES

AddTwoInts.srv

)

## Generate actions in the 'action' folder //在“action”文件夹中生成操作

# add\_action\_files(

# FILES

# Action1.action

# Action2.action

# )

## Generate added messages and services with any dependencies listed here

## 使用此处列出的任何依赖项生成添加的消息和服务

generate\_messages(

DEPENDENCIES

std\_msgs

)

################################################

## Declare ROS dynamic reconfigure parameters ##

##声明ROS动态重新配置参数##

################################################

## To declare and build dynamic reconfigure parameters within this package, follow these steps:

##要在此程序包中声明和构建动态重新配置参数，请按照下列步骤操作：

## \* In the file package.xml:

## \* add a build\_depend and a run\_depend tag for "dynamic\_reconfigure"

## 为“dynamic\_reconfigure”添加一个build\_depend和一个run\_depend标记，

## \* In this file (CMakeLists.txt):

## \* add "dynamic\_reconfigure" to

## find\_package(catkin REQUIRED COMPONENTS ...)

## 添加“dynamic\_reconfigure” 到find\_package（catkin所需组件...）

## \* uncomment the "generate\_dynamic\_reconfigure\_options" section below

## and list every .cfg file to be processed

## 取消注释下面的“generate\_dynamic\_reconfigure\_options”部分，并列出要处理的每个.cfg文件

## Generate dynamic reconfigure parameters in the 'cfg' folder

##在“cfg”文件夹中生成动态重新配置参数

# generate\_dynamic\_reconfigure\_options(

# cfg/DynReconf1.cfg

# cfg/DynReconf2.cfg

# )

###################################

## catkin specific configuration ##

## catkin具体配置##

###################################

## The catkin\_package macro generates cmake config files for your package

##　 catkin\_package宏为您的包生成cmake配置文件

## Declare things to be passed to dependent projects

##　声明要传递给依赖项目的东西

## INCLUDE\_DIRS: uncomment this if you package contains header files

## INCLUDE\_DIRS：如果程序包包含头文件，请取消注释

## LIBRARIES: libraries you create in this project that dependent projects also need

## 库：您在此项目中创建的库，依赖项目也需要

## CATKIN\_DEPENDS: catkin\_packages dependent projects also need

## CATKIN DEPENDS：catkin包依赖项目也需要

## DEPENDS: system dependencies of this project that dependent projects also need

## DEPENDS：依赖项目还需要的此项目的系统依赖性

catkin\_package(

# INCLUDE\_DIRS include

# LIBRARIES beginner\_tutorials

# CATKIN\_DEPENDS roscpp rospy std\_msgs

CATKIN\_DEPENDS message\_runtime

# DEPENDS system\_lib

)

###########

## Build ##

###########

## Specify additional locations of header files

##指定头文件的其他位置

## Your package locations should be listed before other locations

## 您的软件包位置应该在其他位置之前列出

# include\_directories(include)

include\_directories(

${catkin\_INCLUDE\_DIRS}

)

## Declare a C++ library

# add\_library(beginner\_tutorials

# src/${PROJECT\_NAME}/beginner\_tutorials.cpp

# )

## Add cmake target dependencies of the library

## 添加库的cmake目标依赖关系

## as an example, code may need to be generated before libraries

## either from message generation or dynamic reconfigure

##作为示例，可能需要在来自消息生成或动态重新配置的库之前生成代码

# add\_dependencies(beginner\_tutorials ${${PROJECT\_NAME}\_EXPORTED\_TARGETS} ${catkin\_EXPORTED\_TARGETS})

## Declare a C++ executable

# add\_executable(beginner\_tutorials\_node src/beginner\_tutorials\_node.cpp)

add\_executable(talker src/talker.cpp)

add\_executable(listener src/listener.cpp)

add\_executable(add\_two\_ints\_server src/add\_two\_ints\_server.cpp)

add\_executable(add\_two\_ints\_client src/add\_two\_ints\_client.cpp)

## Add cmake target dependencies of the executable

## same as for the library above

# add\_dependencies(beginner\_tutorials\_node ${${PROJECT\_NAME}\_EXPORTED\_TARGETS} ${catkin\_EXPORTED\_TARGETS})

add\_dependencies(talker ${${PROJECT\_NAME}\_generate\_message\_cpp} ${catkin\_EXPORTED\_TARGETS})

add\_dependencies(listener ${${PROJECT\_NAME}\_generate\_message\_cpp} ${catkin\_EXPORTED\_TARGETS})

add\_dependencies(add\_two\_ints\_server ${${PROJECT\_NAME}\_generate\_message\_cpp} ${catkin\_EXPORTED\_TARGETS})

add\_dependencies(add\_two\_ints\_client ${${PROJECT\_NAME}\_generate\_message\_cpp} ${catkin\_EXPORTED\_TARGETS})

## Specify libraries to link a library or executable target against

## 指定要链接库或可执行目标的库

# target\_link\_libraries(beginner\_tutorials\_node

# ${catkin\_LIBRARIES}

# )

target\_link\_libraries(talker

${catkin\_LIBRARIES}

)

target\_link\_libraries(listener

${catkin\_LIBRARIES}

)

target\_link\_libraries(add\_two\_ints\_server

${catkin\_LIBRARIES}

)

target\_link\_libraries(add\_two\_ints\_client

${catkin\_LIBRARIES}

)

#############

## Install ##

#############

# all install targets should use catkin DESTINATION variables

# See http://ros.org/doc/api/catkin/html/adv\_user\_guide/variables.html

## Mark executable scripts (Python etc.) for installation

## in contrast to setup.py, you can choose the destination

# install(PROGRAMS

# scripts/my\_python\_script

# DESTINATION ${CATKIN\_PACKAGE\_BIN\_DESTINATION}

# )

## Mark executables and/or libraries for installation

# install(TARGETS beginner\_tutorials beginner\_tutorials\_node

# ARCHIVE DESTINATION ${CATKIN\_PACKAGE\_LIB\_DESTINATION}

# LIBRARY DESTINATION ${CATKIN\_PACKAGE\_LIB\_DESTINATION}

# RUNTIME DESTINATION ${CATKIN\_PACKAGE\_BIN\_DESTINATION}

# )

## Mark cpp header files for installation

# install(DIRECTORY include/${PROJECT\_NAME}/

# DESTINATION ${CATKIN\_PACKAGE\_INCLUDE\_DESTINATION}

# FILES\_MATCHING PATTERN "\*.h"

# PATTERN ".svn" EXCLUDE

# )

## Mark other files for installation (e.g. launch and bag files, etc.)

# install(FILES

# # myfile1

# # myfile2

# DESTINATION ${CATKIN\_PACKAGE\_SHARE\_DESTINATION}

# )

#############

## Testing ##

#############

## Add gtest based cpp test target and link libraries

# catkin\_add\_gtest(${PROJECT\_NAME}-test test/test\_beginner\_tutorials.cpp)

# if(TARGET ${PROJECT\_NAME}-test)

# target\_link\_libraries(${PROJECT\_NAME}-test ${PROJECT\_NAME})

# endif()

## Add folders to be run by python nosetests

# catkin\_add\_nosetests(test)