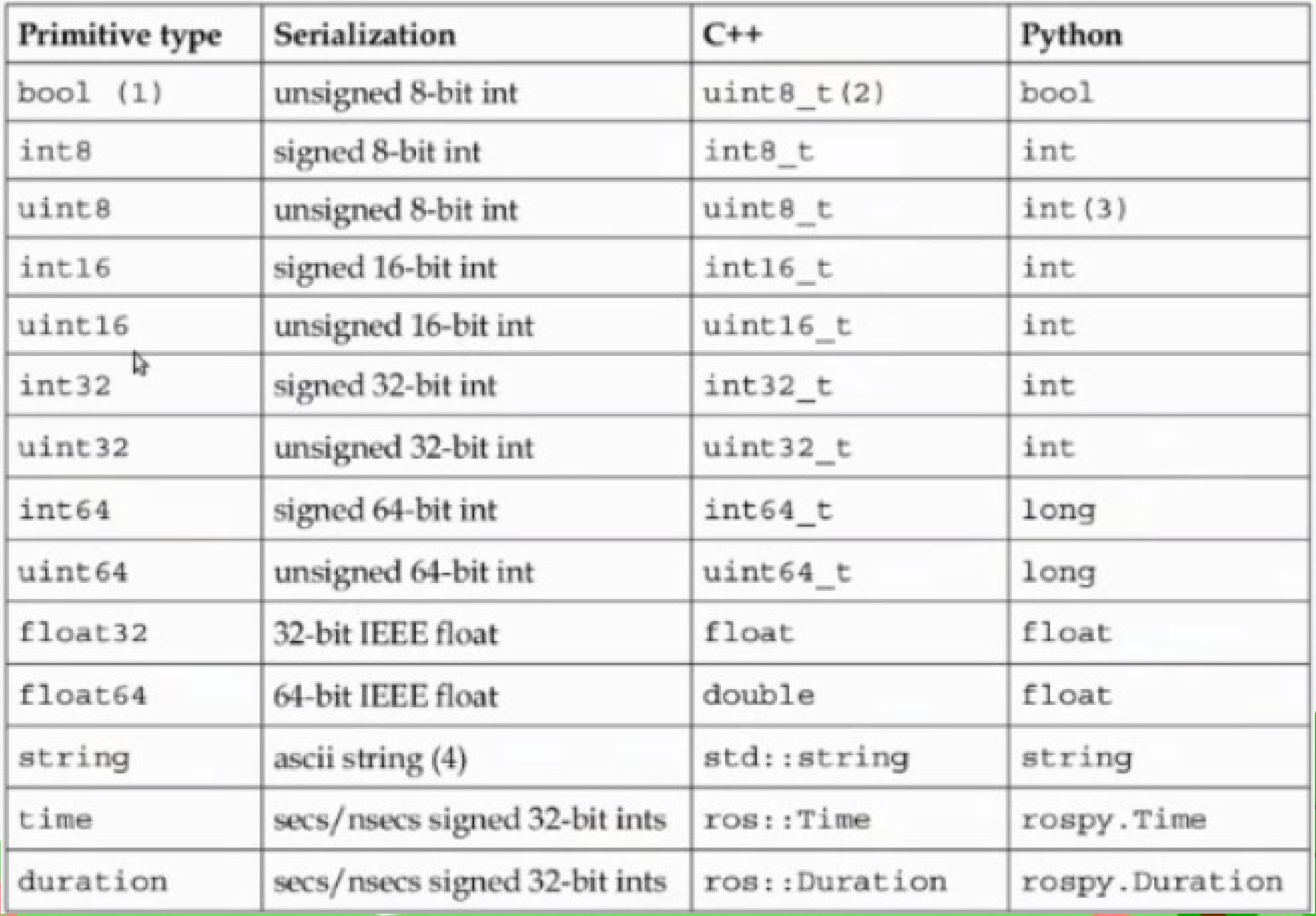
Robot Operating System

Chapter 2

Intro to Sevices, Action & Custom message

Pyae Soan Aung
RHCSA
Rom Robitics Co.ltd

Data type



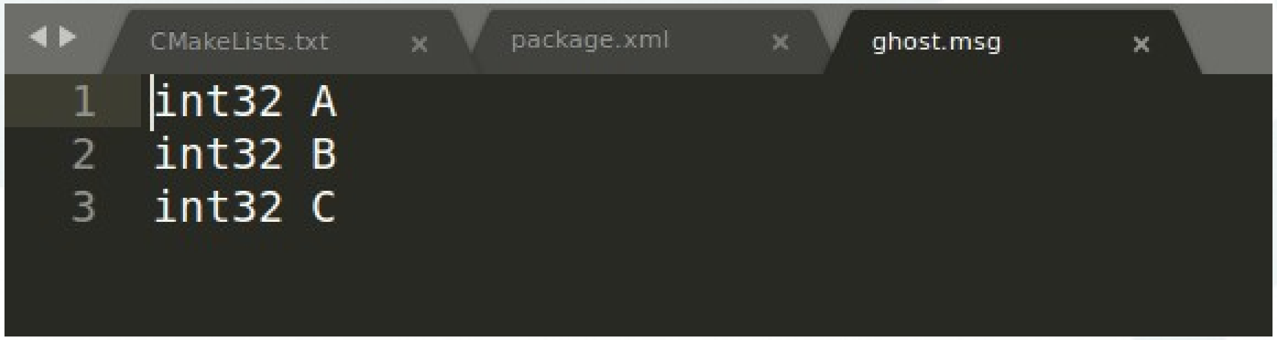
Primitive type	Serialization	C++	Python
bool (1)	unsigned 8-bit int	uint8_t(2)	bool
int8	signed 8-bit int	int8_t	int
uint8	unsigned 8-bit int	uint8_t	int(3)
int16	signed 16-bit int	int16_t	int
uint16	unsigned 16-bit int	uint16_t	int
int32	signed 32-bit int	int32_t	int
uint32	unsigned 32-bit int	uint32_t	int
int64	signed 64-bit int	int64_t	long
uint64	unsigned 64-bit int	uint64_t	long
float32	32-bit IEEE float	float	float
float64	64-bit IEEE float	double	float
string	ascii string (4)	std::string	string
time	secs/nsecs signed 32-bit ints	ros::Time	rospy.Time
duration	secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration

Custom message ဘယ်လိုဖန်တီးမလဲ?

```
~$ cd your/path/catkin_ws/src

~$ catkin_create_pkg psa_server roscpp rospy std_msgs message_generation

~$ catkin_make

~$ roscd psa_server

~$ mkdir msg

~$ subl msg/ghost.msg

~$ cat msg/ghost.msg
```

ပြီးလျှင် Header များထွက်လာအောင် compile လုပ်ပါ။

Creating custom message


CMakeLists.txt package.xml ghost.msg

```
int32 A
int32 B
int32 C
```

Package ထဲက CmakeLists.txt နဲ့ package.xml တို့ကိုပြင်ပေးဖို့လိုပါလိမ့်မယ်။

CmakeLists.txt

```
find_package(catkin REQUIRED COMPONENTS
        ..
        message_generation)
add_message_flies(FILES
        ..
        ghost.msg)
generate_message(
        DEPENDENCIES
        std_msgs)
```

Package.xml

```
<biuld_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

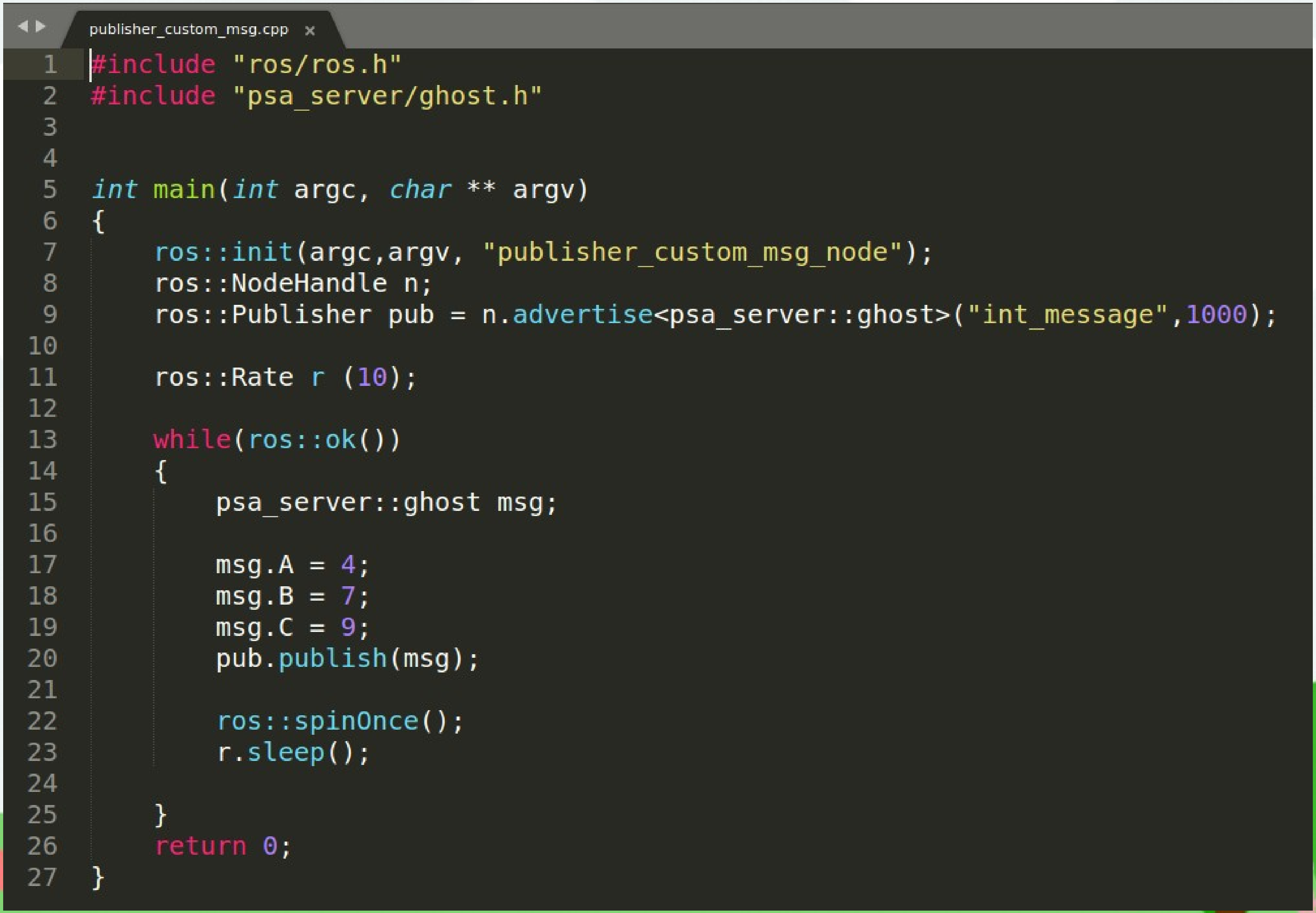
How to check?

```
~$ rosmsg show psa_server/ghost
```

Where is Header ?

Using custom message

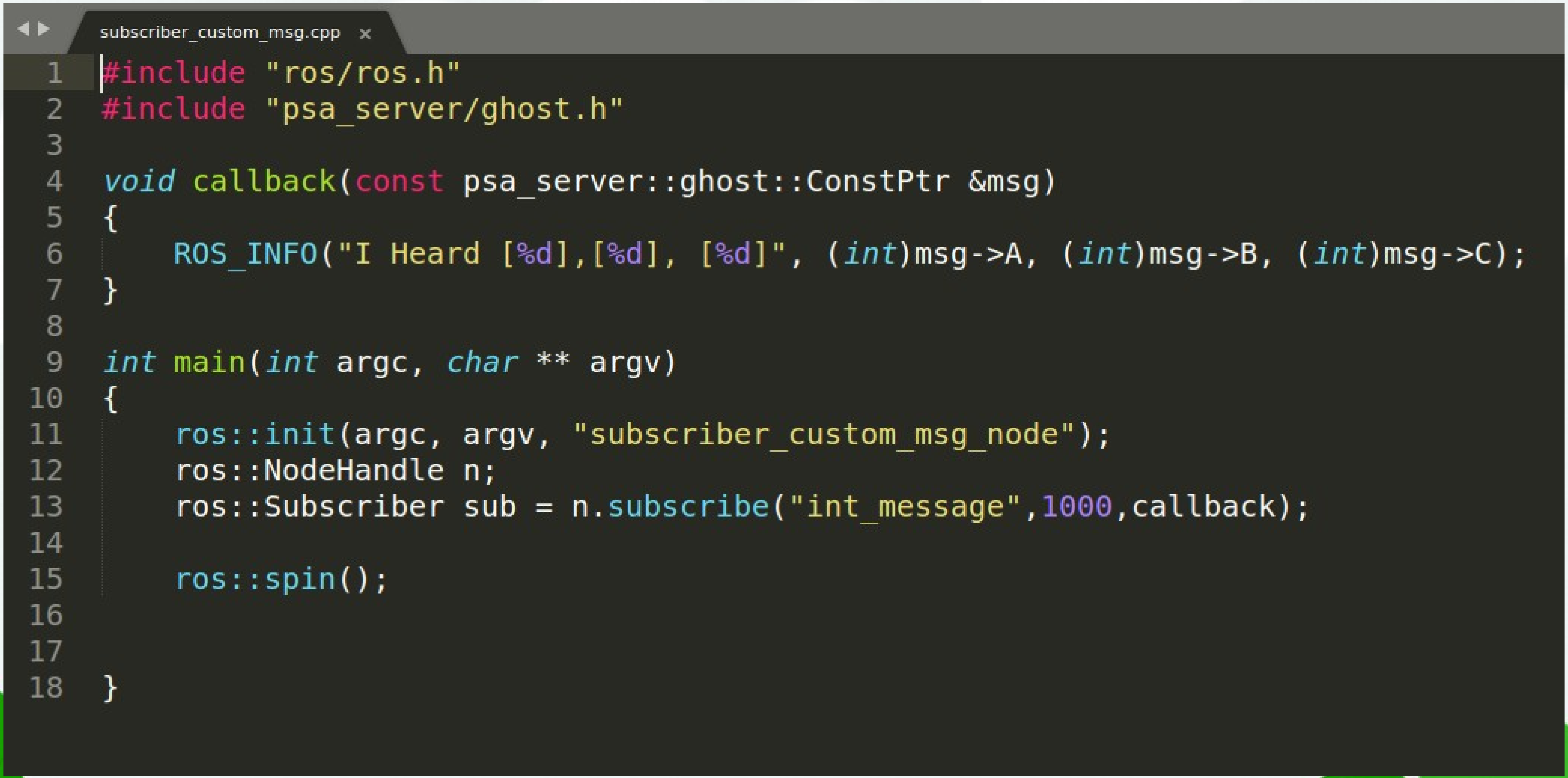
Publisher node


publisher_custom_msg.cpp

```cpp
#include "ros/ros.h"
#include "psa_server/ghost.h"


int main(int argc, char ** argv)
{
    ros::init(argc,argv, "publisher_custom_msg_node");
    ros::NodeHandle n;
    ros::Publisher pub = n.advertise<psa_server::ghost>("int_message",1000);

    ros::Rate r (10);

    while(ros::ok())
    {
        psa_server::ghost msg;

        msg.A = 4;
        msg.B = 7;
        msg.C = 9;
        pub.publish(msg);

        ros::spinOnce();
        r.sleep();

    }
    return 0;
}
```

Using custom message

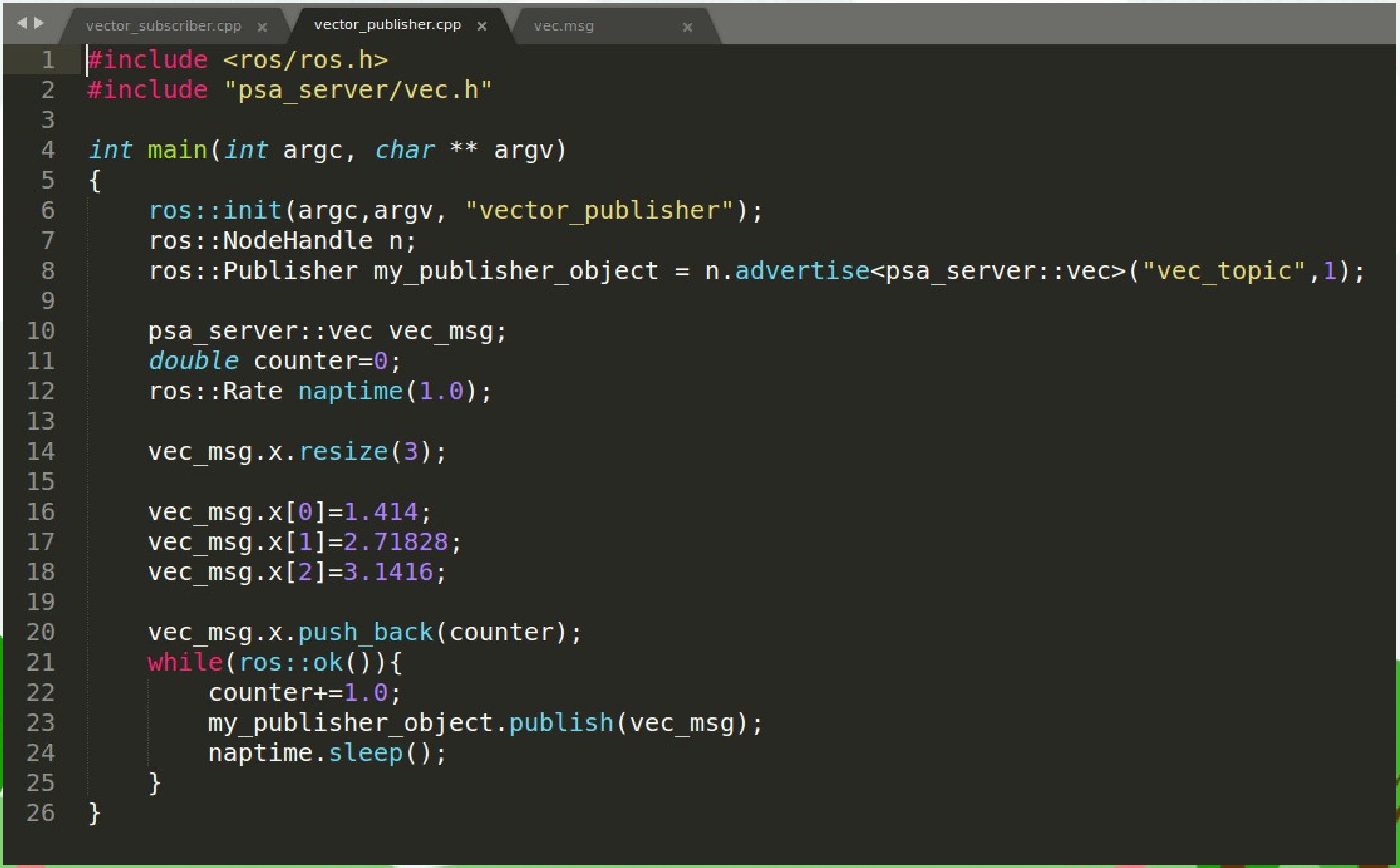
Subscriber node



```cpp
#include "ros/ros.h"
#include "psa_server/ghost.h"

void callback(const psa_server::ghost::ConstPtr &msg)
{
    ROS_INFO("I Heard [%d],[%d], [%d]", (int)msg->A, (int)msg->B, (int)msg->C);
}

int main(int argc, char ** argv)
{
    ros::init(argc, argv, "subscriber_custom_msg_node");
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("int_message",1000,callback);

    ros::spin();


}
```

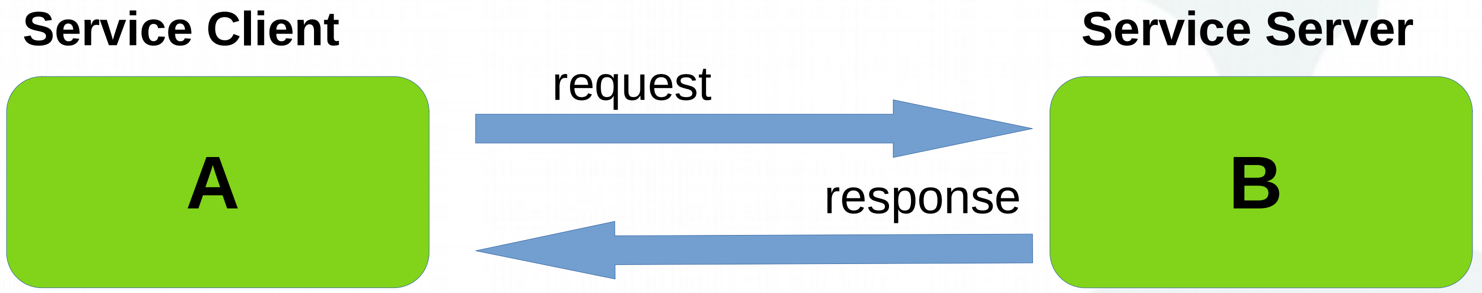
Using custom message

Sending vector



```cpp
#include <ros/ros.h>
#include "psa_server/vec.h"

int main(int argc, char ** argv)
{
    ros::init(argc,argv, "vector_publisher");
    ros::NodeHandle n;
    ros::Publisher my_publisher_object = n.advertise<psa_server::vec>("vec_topic",1);

    psa_server::vec vec_msg;
    double counter=0;
    ros::Rate naptime(1.0);

    vec_msg.x.resize(3);

    vec_msg.x[0]=1.414;
    vec_msg.x[1]=2.71828;
    vec_msg.x[2]=3.1416;

    vec_msg.x.push_back(counter);
    while(ros::ok()){
        counter+=1.0;
        my_publisher_object.publish(vec_msg);
        naptime.sleep();
    }
}
```

Communication method: Service



Client.call(service object)

Server Callback function

🟩 = node

How to check?

Use tab key

```
~$ rossrv show psa_server/ghostsrv
```

Where is Header ?

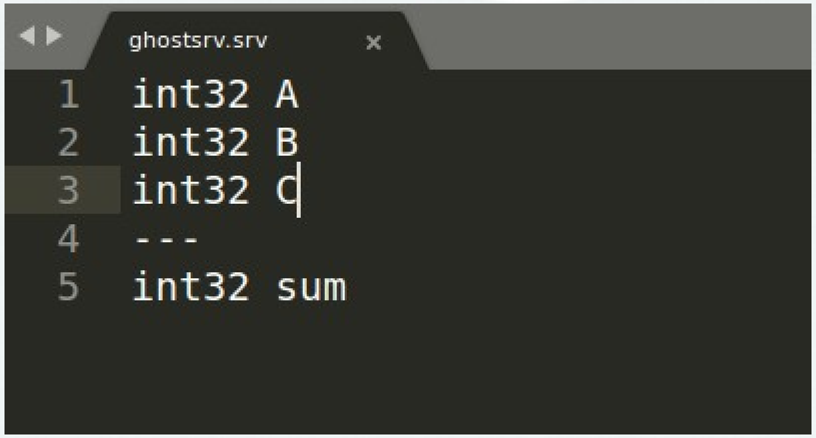
How to create service

~$ roscd psa_server

~$ mkdir srv

~$ subl srv/ghostsrv.srv

~$ cat srv/ghostsrv.msg



```
int32 A
int32 B
int32 C
---
int32 sum
```

ပြီးရင် compile လုပ်ပါ။

How to create service

Package ထဲက CmakeLists.txt နဲ့ package.xml တို့ကိုပြင်ပေးဖို့လိုပါလိမ့်မယ်။

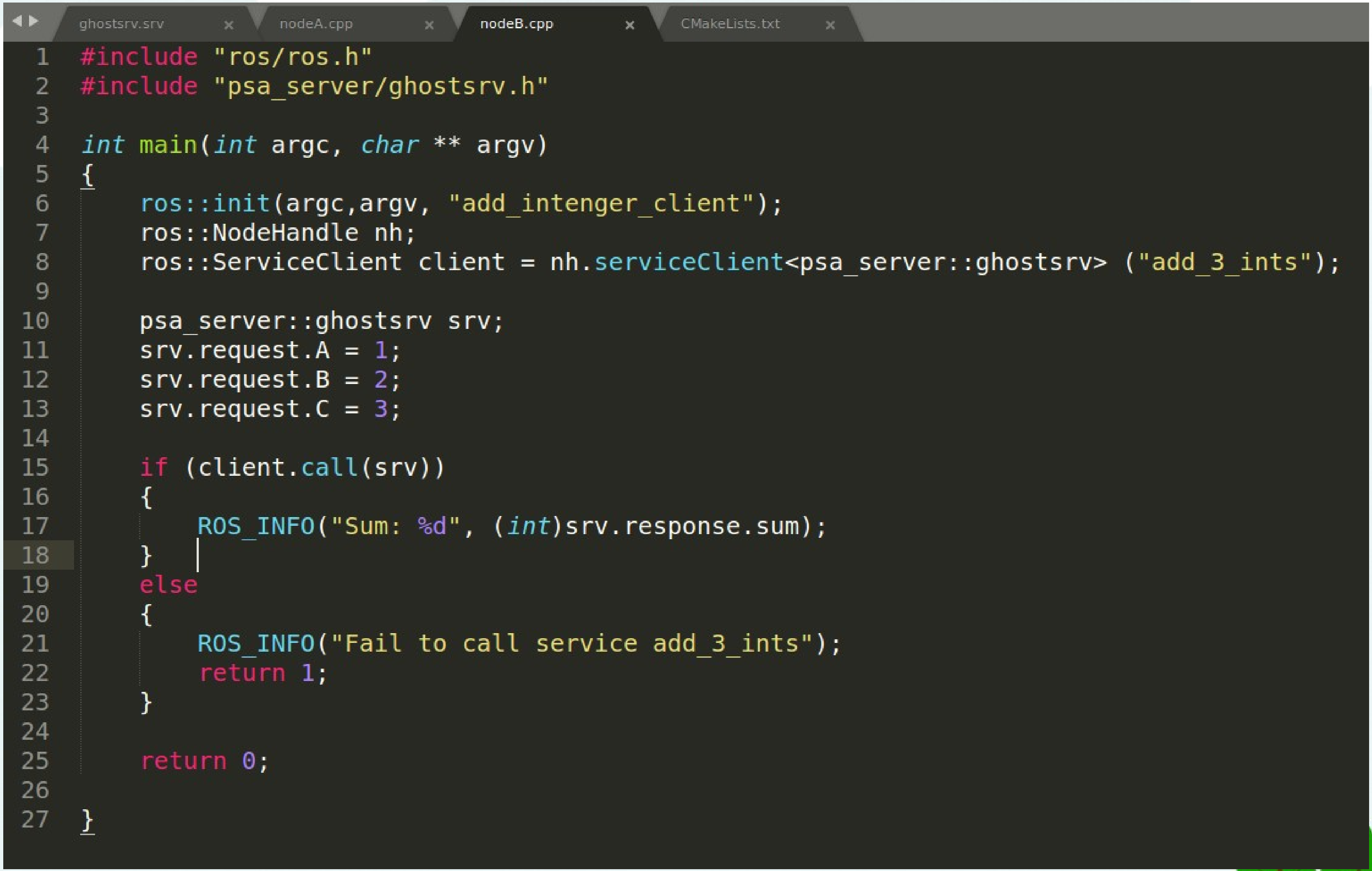
CmakeLists.txt

```
find_package(catkin REQUIRED COMPONENTS
        ..
        message_generation)
add_service_flies(FILES
        ..
        ghostsrv.srv)
generate_message(
        DEPENDENCIES
        std_msgs)
```

Package.xml

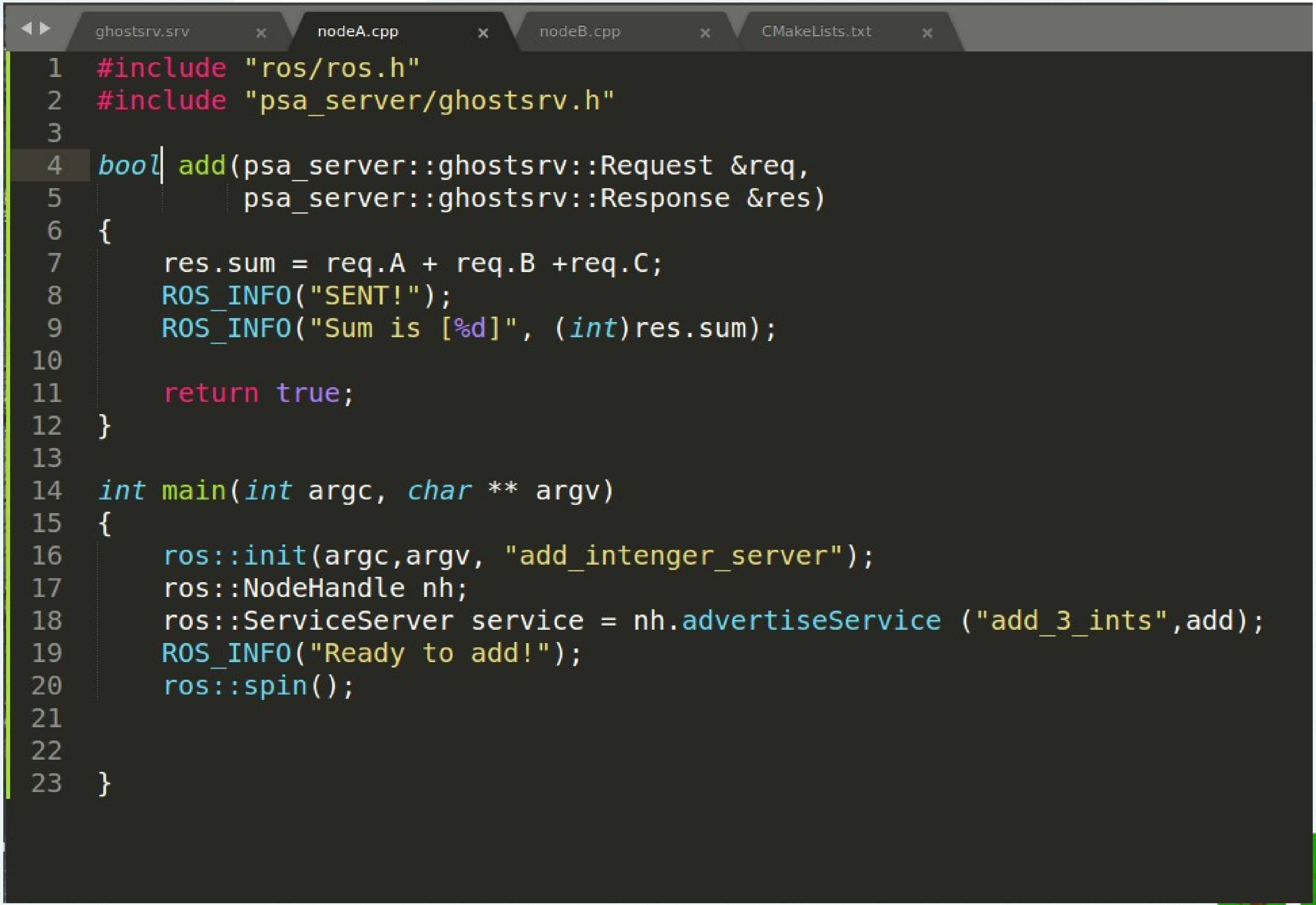
```
<biuld_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>
```

Service Client



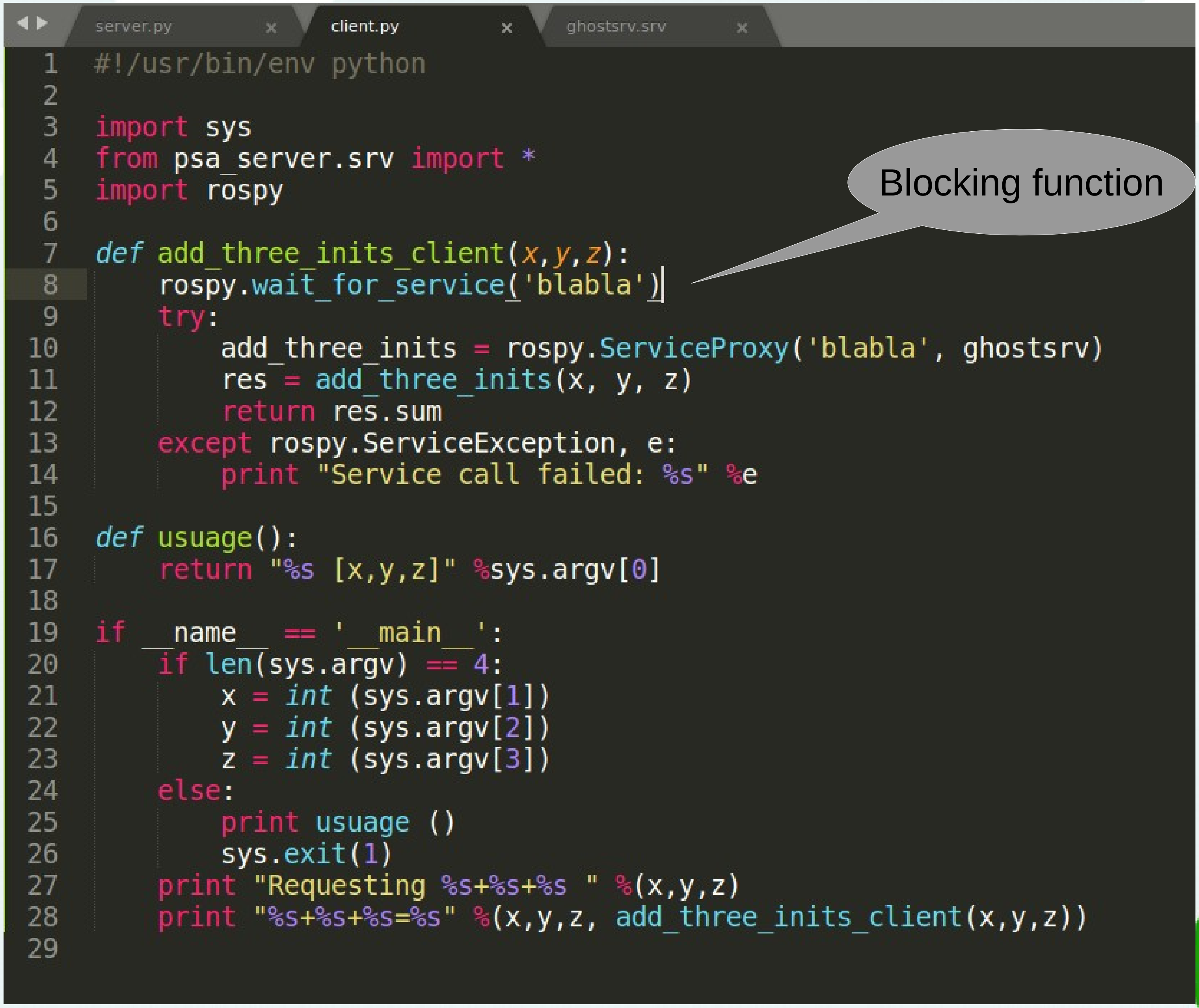
```cpp
#include "ros/ros.h"
#include "psa_server/ghostsrv.h"

int main(int argc, char ** argv)
{
    ros::init(argc,argv, "add_intenger_client");
    ros::NodeHandle nh;
    ros::ServiceClient client = nh.serviceClient<psa_server::ghostsrv> ("add_3_ints");

    psa_server::ghostsrv srv;
    srv.request.A = 1;
    srv.request.B = 2;
    srv.request.C = 3;

    if (client.call(srv))
    {
        ROS_INFO("Sum: %d", (int)srv.response.sum);
    }
    else
    {
        ROS_INFO("Fail to call service add_3_ints");
        return 1;
    }

    return 0;

}
```

Service Server



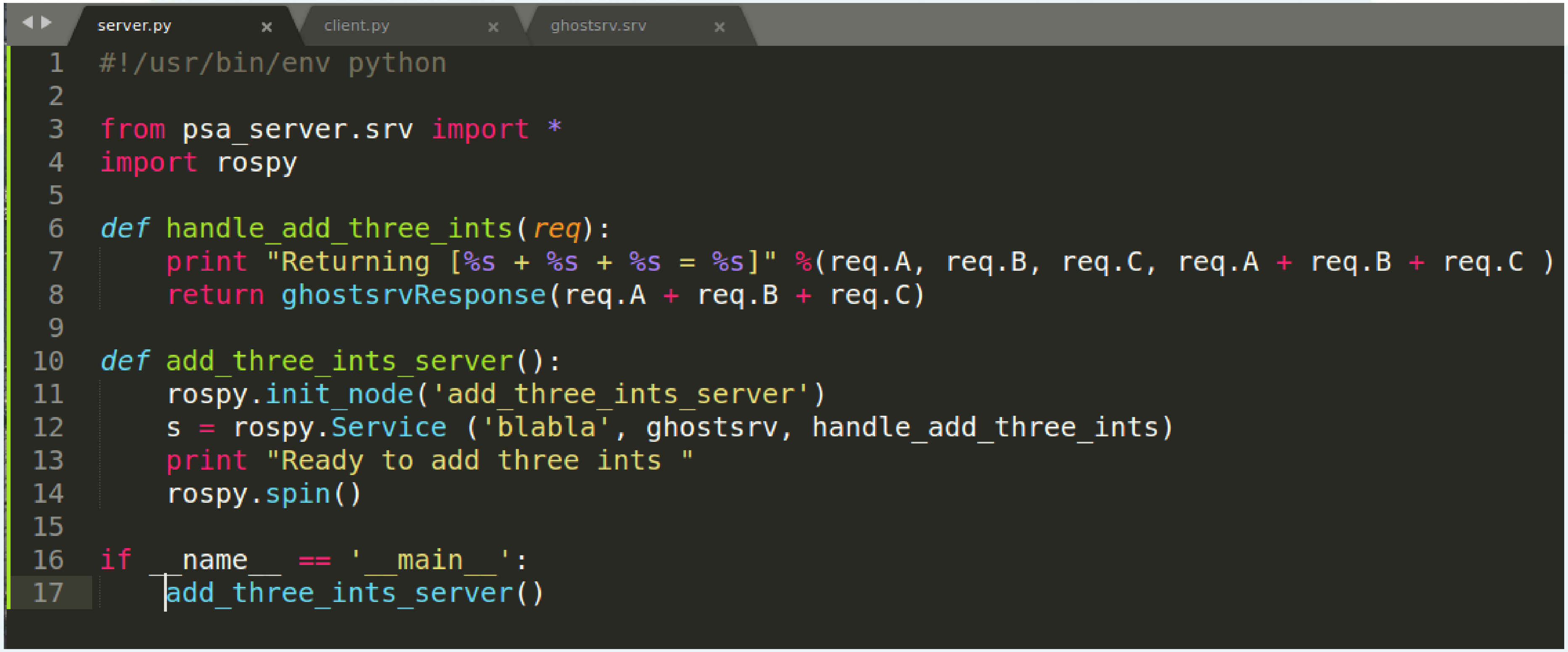
```cpp
#include "ros/ros.h"
#include "psa_server/ghostsrv.h"

bool add(psa_server::ghostsrv::Request &req,
         psa_server::ghostsrv::Response &res)
{
    res.sum = req.A + req.B +req.C;
    ROS_INFO("SENT!");
    ROS_INFO("Sum is [%d]", (int)res.sum);

    return true;
}

int main(int argc, char ** argv)
{
    ros::init(argc,argv, "add_intenger_server");
    ros::NodeHandle nh;
    ros::ServiceServer service = nh.advertiseService ("add_3_ints",add);
    ROS_INFO("Ready to add!");
    ros::spin();


}
```

Service Client (python)



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

import sys
from psa_server.srv import *
import rospy

def add_three_inits_client(x,y,z):
    rospy.wait_for_service('blabla')
    try:
        add_three_inits = rospy.ServiceProxy('blabla', ghostsrv)
        res = add_three_inits(x, y, z)
        return res.sum
    except rospy.ServiceException, e:
        print "Service call failed: %s" %e

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

if __name__ == '__main__':
    if len(sys.argv) == 4:
        x = int (sys.argv[1])
        y = int (sys.argv[2])
        z = int (sys.argv[3])
    else:
        print usuage ()
        sys.exit(1)
    print "Requesting %s+%s+%s " %(x,y,z)
    print "%s+%s+%s=%s" %(x,y,z, add_three_inits_client(x,y,z))

```

Service Server (python)

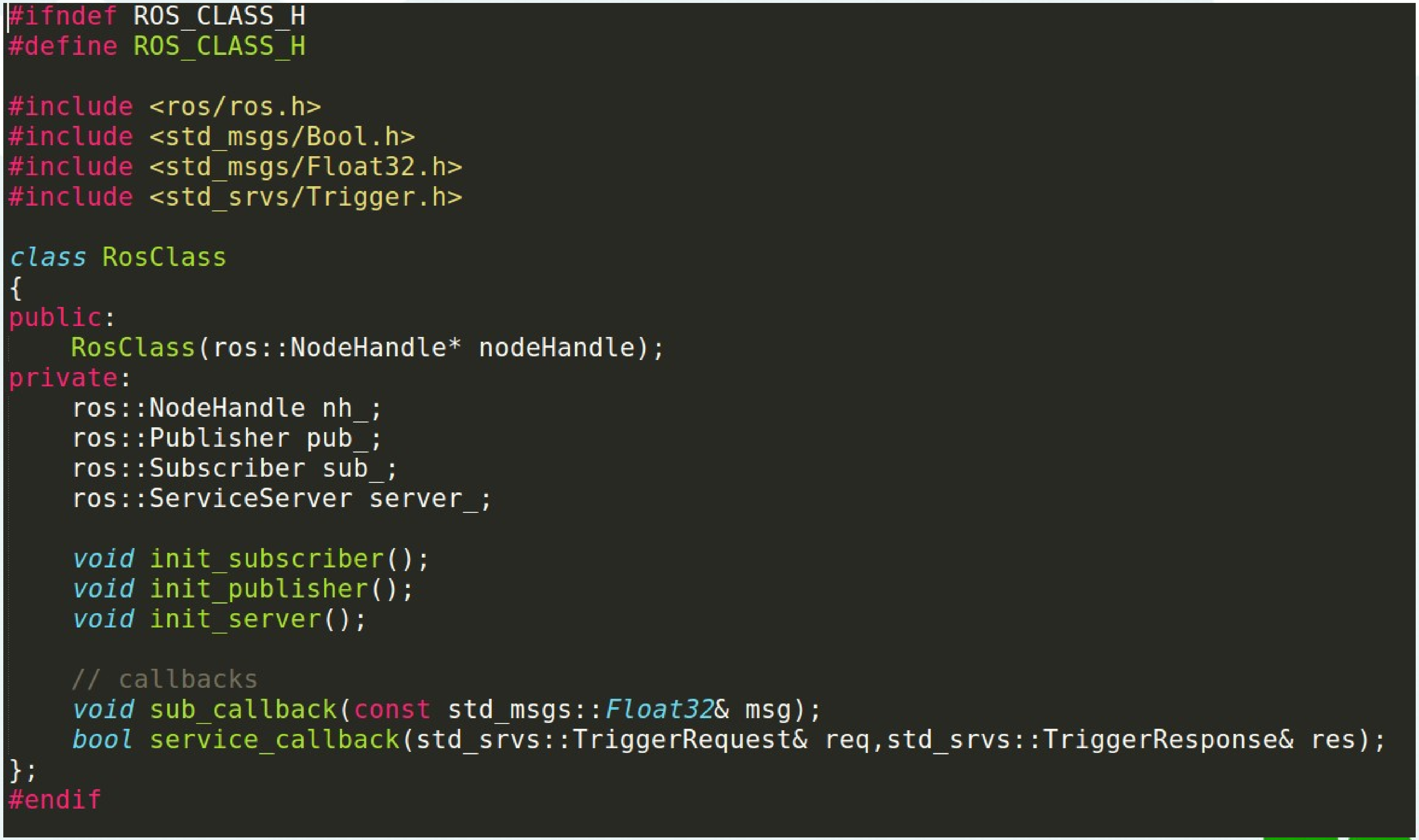


```python
#!/usr/bin/env python

from psa_server.srv import *
import rospy

def handle_add_three_ints(req):
    print "Returning [%s + %s + %s = %s]" %(req.A, req.B, req.C, req.A + req.B + req.C )
    return ghostsrvResponse(req.A + req.B + req.C)

def add_three_ints_server():
    rospy.init_node('add_three_ints_server')
    s = rospy.Service ('blabla', ghostsrv, handle_add_three_ints)
    print "Ready to add three ints "
    rospy.spin()

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

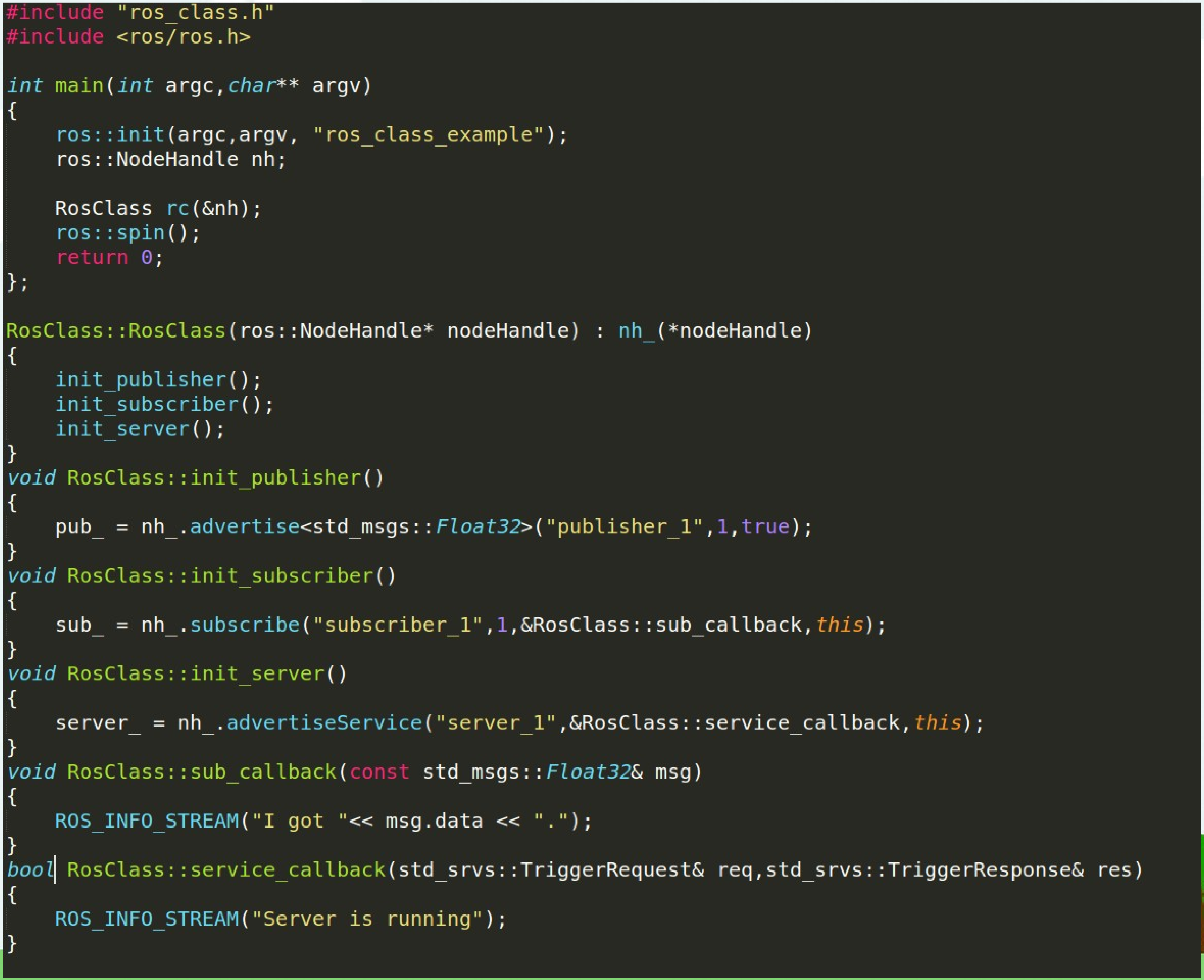
Topic တွေကို rostopic နဲ့ စစ်လို့ရသလို Service တွေကိုလည်း rosservice နဲ့စစ်ဆေးလို့ရပါတယ်။

Message တွေကြည့်ချင်တဲ့အခါ rosmsg နဲ့ ကြည့်နိုင်သလို service မှာလည်း rossrv နဲ့ကြည့်နိုင်ပါတယ်။

Using C++ Class with ROS

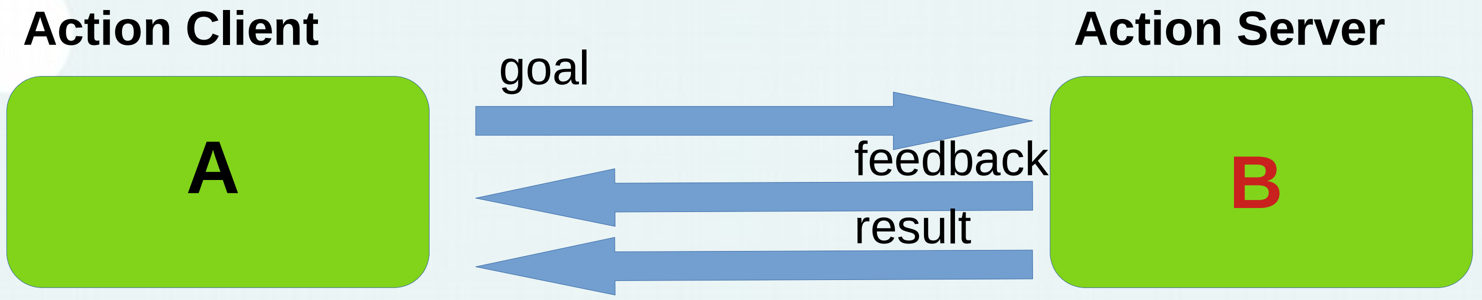


```
#ifndef ROS_CLASS_H
#define ROS_CLASS_H

#include <ros/ros.h>
#include <std_msgs/Bool.h>
#include <std_msgs/Float32.h>
#include <std_srvs/Trigger.h>

class RosClass
{
public:
    RosClass(ros::NodeHandle* nodeHandle);
private:
    ros::NodeHandle nh_;
    ros::Publisher pub_;
    ros::Subscriber sub_;
    ros::ServiceServer server_;

    void init_subscriber();
    void init_publisher();
    void init_server();

    // callbacks
    void sub_callback(const std_msgs::Float32& msg);
    bool service_callback(std_srvs::TriggerRequest& req,std_srvs::TriggerResponse& res);
};
#endif
```



```cpp
#include "ros_class.h"
#include <ros/ros.h>

int main(int argc,char** argv)
{
    ros::init(argc,argv, "ros_class_example");
    ros::NodeHandle nh;

    RosClass rc(&nh);
    ros::spin();
    return 0;
};

RosClass::RosClass(ros::NodeHandle* nodeHandle) : nh_(*nodeHandle)
{
    init_publisher();
    init_subscriber();
    init_server();
}
void RosClass::init_publisher()
{
    pub_ = nh_.advertise<std_msgs::Float32>("publisher_1",1,true);
}
void RosClass::init_subscriber()
{
    sub_ = nh_.subscribe("subscriber_1",1,&RosClass::sub_callback,this);
}
void RosClass::init_server()
{
    server_ = nh_.advertiseService("server_1",&RosClass::service_callback,this);
}
void RosClass::sub_callback(const std_msgs::Float32& msg)
{
    ROS_INFO_STREAM("I got "<< msg.data << ".");
}
bool RosClass::service_callback(std_srvs::TriggerRequest& req,std_srvs::TriggerResponse& res)
{
    ROS_INFO_STREAM("Server is running");
}
```

Communication method: Action



```
client.waitForServer()
Client.sendGoal(goal object)
Client.waitForResult(Duration)
Client.getState()
Client.cancelAllGoals()
Client.cancelGoal()
```

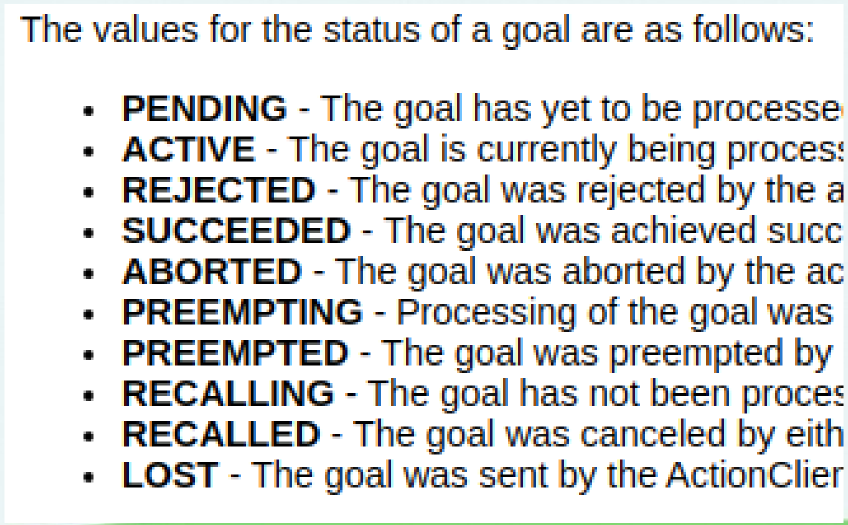
ActionServer (node,name,boost::function,bool auto_start)

Server.start()

Server.registerPreemptCallback()

Server.setAborted()

Server.setSucceded()


The values for the status of a goal are as follows:

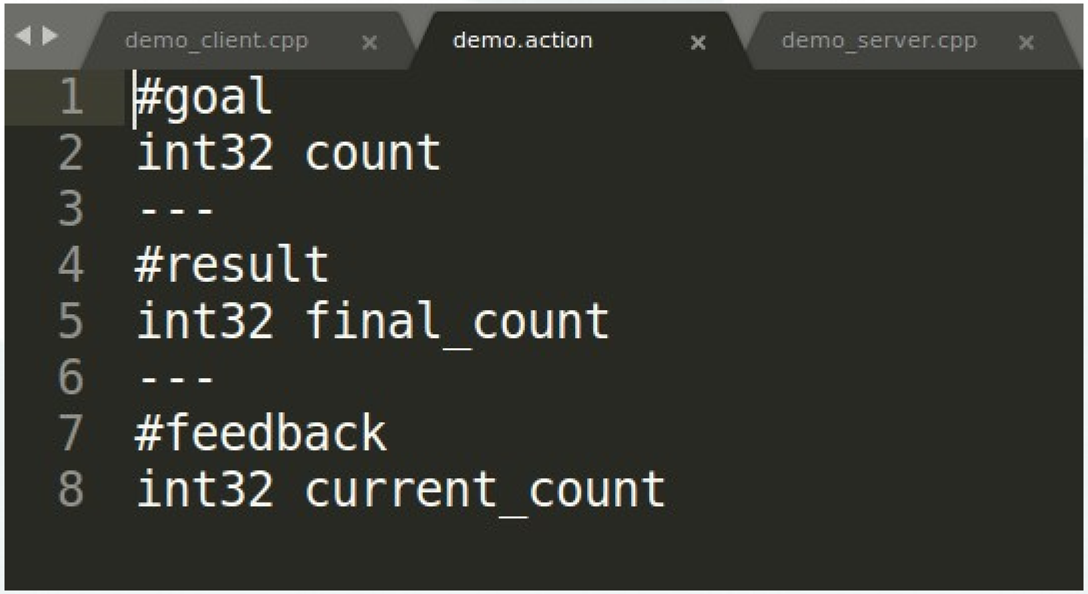
- **PENDING** - The goal has yet to be processe
- **ACTIVE** - The goal is currently being proces
- **REJECTED** - The goal was rejected by the a
- **SUCCEEDED** - The goal was achieved succ
- **ABORTED** - The goal was aborted by the ac
- **PREEMPTING** - Processing of the goal was
- **PREEMPTED** - The goal was preempted by
- **RECALLING** - The goal has not been proces
- **RECALLED** - The goal was canceled by eith
- **LOST** - The goal was sent by the ActionClier

https://docs.ros.org/diamondback/api/actionlib/html/index.html

Action တစ်ခု ဘယ်လိုဖန်တီးမလဲ?

Package ဖန်တီးသည့်အခါ dependency အဖြစ် actionlib နဲ့ actionlib_msgs ကိုထည့်ပါ။

```
~$ roscd psa_server

~$ mkdir action

~$ subl action/demo.action

~$ cat msg/ghost.msg
```



```
#goal
int32 count
---
#result
int32 final_count
---
#feedback
int32 current_count
```

ပြီးလျှင် Header များထွက်လာအောင် compile လုပ်ပါ။

Creating Action

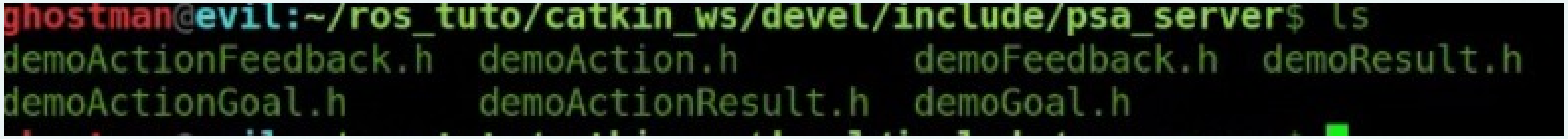
CmakeLists.txt

```
    find_package(catkin REQUIRED COMPONENTS
            ..
            message_generation
            actionlib
            actionlib_msgs)
    add_action_flies(FILES
            ..
            demo.action)
    generate_message(
            DEPENDENCIES
            std_msgs
            actionlib_msgs)
    catkin_package(
            CATKIN_DEPENDS actionlib roscpp rospy std_msgs actionlib_msgs)
    include_directories(include
            ...
            ${Boost_INCLUDE_DIRS})
```

Package.xml

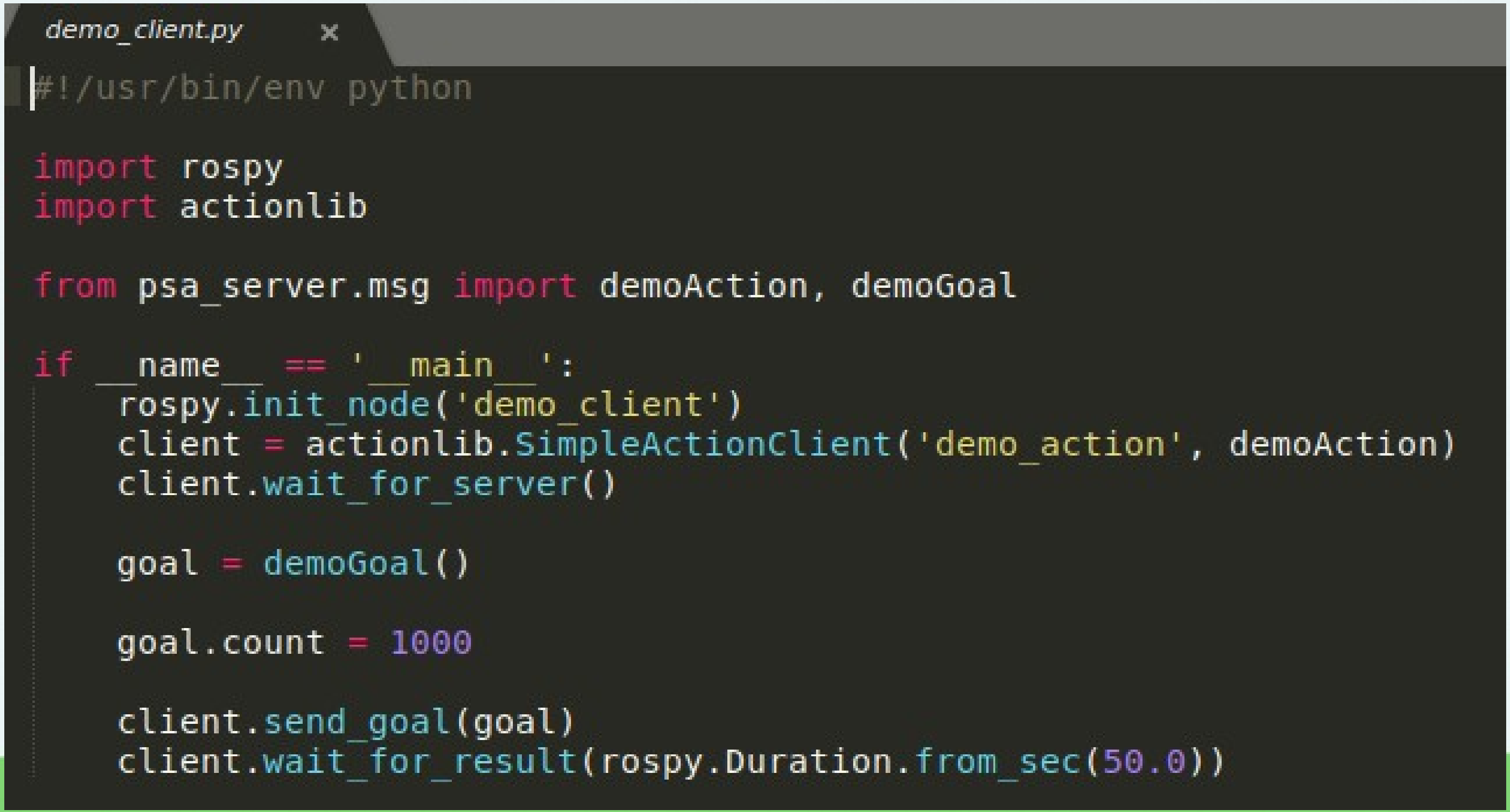
```
            <biuld_depend>message_generation</build_depend>
            <exec_depend>message_runtime</exec_depend>
```

ပြီးရင် compile လုပ်ပြီး header ဖိုင်များကို အသုံးပြုလို့ရပါပြီ



```
ghostman@evil:~/ros_tuto/catkin_ws/devel/include/psa_server$ ls
demoActionFeedback.h  demoAction.h        demoFeedback.h  demoResult.h
demoActionGoal.h      demoActionResult.h  demoGoal.h
```

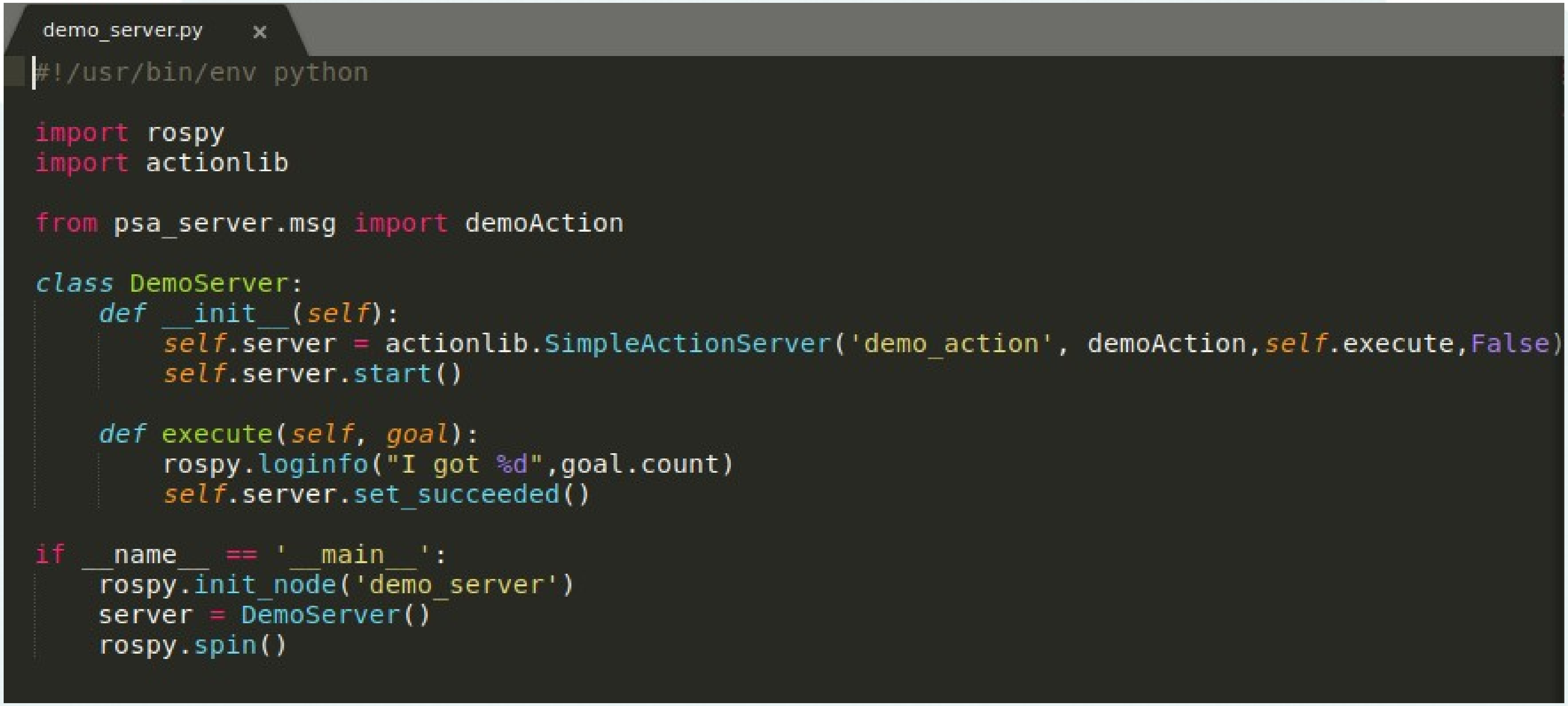
Action client (python)



demo_client.py

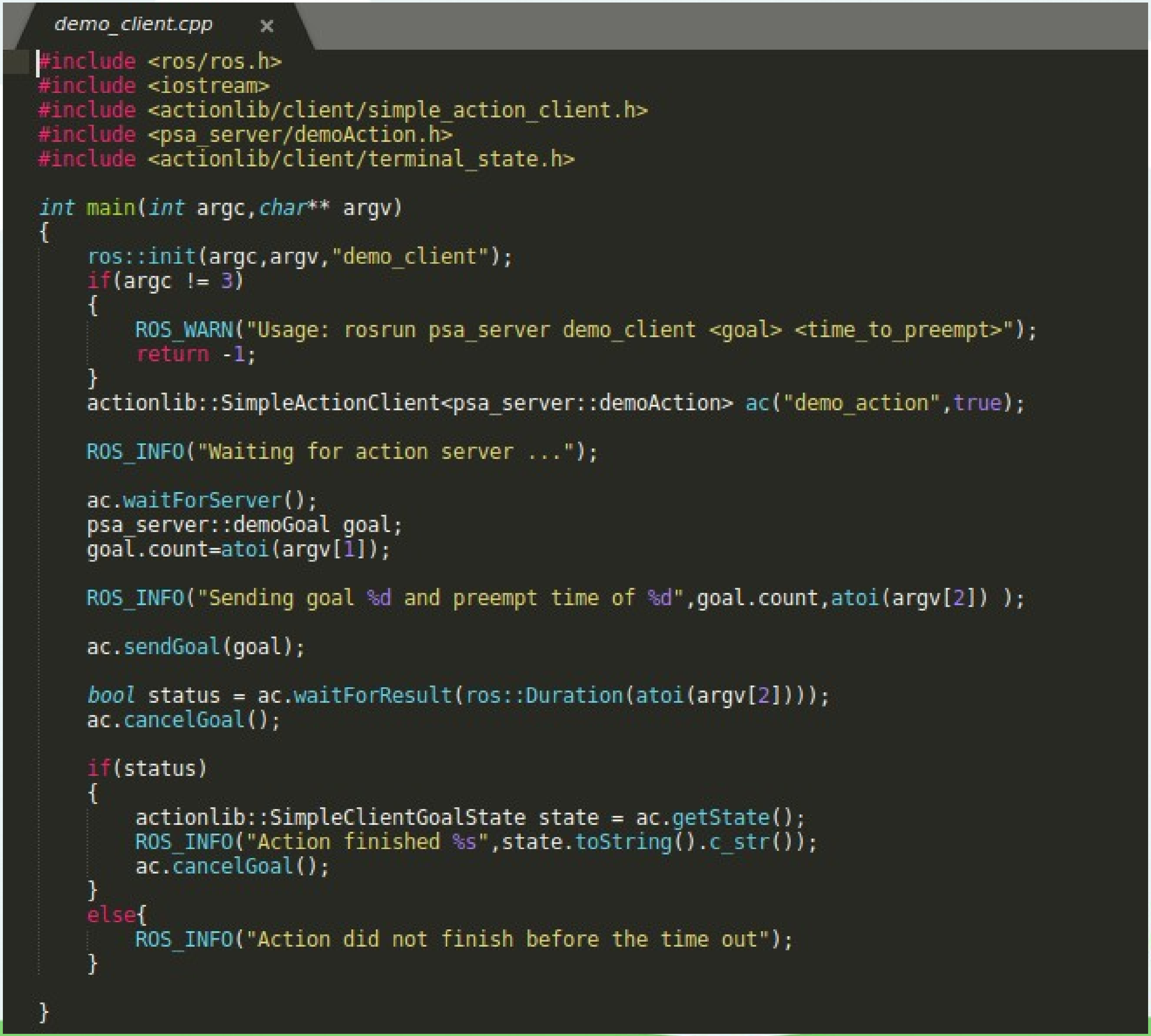
```python
#!/usr/bin/env python

import rospy
import actionlib

from psa_server.msg import demoAction, demoGoal

if __name__ == '__main__':
    rospy.init_node('demo_client')
    client = actionlib.SimpleActionClient('demo_action', demoAction)
    client.wait_for_server()

    goal = demoGoal()

    goal.count = 1000

    client.send_goal(goal)
    client.wait_for_result(rospy.Duration.from_sec(50.0))
```

Action server (python)


demo_server.py

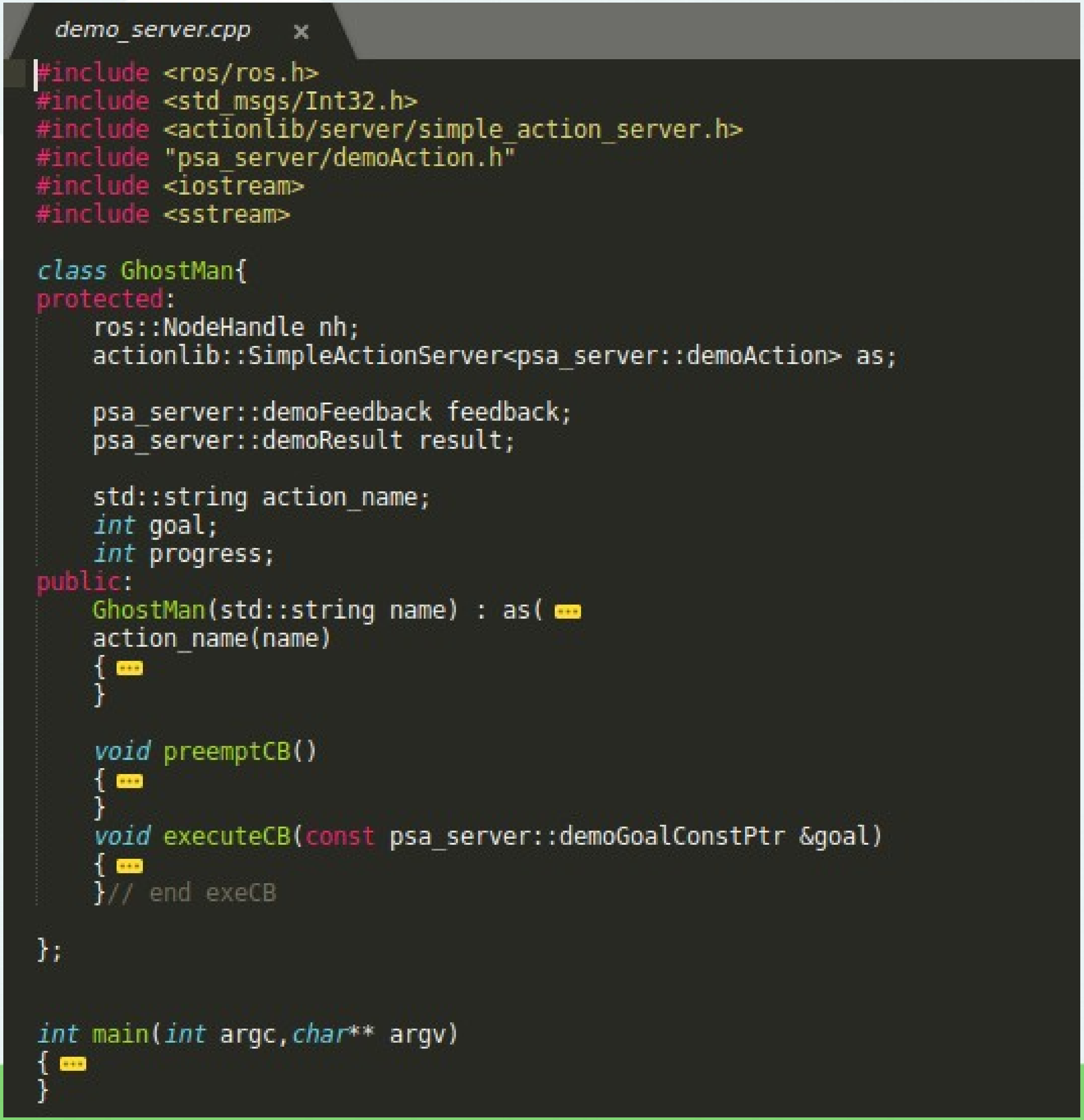
```python
#!/usr/bin/env python

import rospy
import actionlib

from psa_server.msg import demoAction

class DemoServer:
    def __init__(self):
        self.server = actionlib.SimpleActionServer('demo_action', demoAction,self.execute,False)
        self.server.start()

    def execute(self, goal):
        rospy.loginfo("I got %d",goal.count)
        self.server.set_succeeded()

if __name__ == '__main__':
    rospy.init_node('demo_server')
    server = DemoServer()
    rospy.spin()
```

Action client (CPP)


demo_client.cpp

```
#include <ros/ros.h>
#include <iostream>
#include <actionlib/client/simple_action_client.h>
#include <psa_server/demoAction.h>
#include <actionlib/client/terminal_state.h>

int main(int argc,char** argv)
{
    ros::init(argc,argv,"demo_client");
    if(argc != 3)
    {
        ROS_WARN("Usage: rosrun psa_server demo_client <goal> <time_to_preempt>");
        return -1;
    }
    actionlib::SimpleActionClient<psa_server::demoAction> ac("demo_action",true);

    ROS_INFO("Waiting for action server ...");

    ac.waitForServer();
    psa_server::demoGoal goal;
    goal.count=atoi(argv[1]);

    ROS_INFO("Sending goal %d and preempt time of %d",goal.count,atoi(argv[2]) );

    ac.sendGoal(goal);

    bool status = ac.waitForResult(ros::Duration(atoi(argv[2])));
    ac.cancelGoal();

    if(status)
    {
        actionlib::SimpleClientGoalState state = ac.getState();
        ROS_INFO("Action finished %s",state.toString().c_str());
        ac.cancelGoal();
    }
    else{
        ROS_INFO("Action did not finish before the time out");
    }

}
```

Action Server (CPP)


demo_server.cpp

```
#include <ros/ros.h>
#include <std_msgs/Int32.h>
#include <actionlib/server/simple_action_server.h>
#include "psa_server/demoAction.h"
#include <iostream>
#include <sstream>

class GhostMan{
protected:
    ros::NodeHandle nh;
    actionlib::SimpleActionServer<psa_server::demoAction> as;

    psa_server::demoFeedback feedback;
    psa_server::demoResult result;

    std::string action_name;
    int goal;
    int progress;
public:
    GhostMan(std::string name) : as( ...
    action_name(name)
    { ...
    }

    void preemptCB()
    { ...
    }
    void executeCB(const psa_server::demoGoalConstPtr &goal)
    { ...
    }// end exeCB

};


int main(int argc,char** argv)
{ ...
}
```

Thank you!

rom robotics