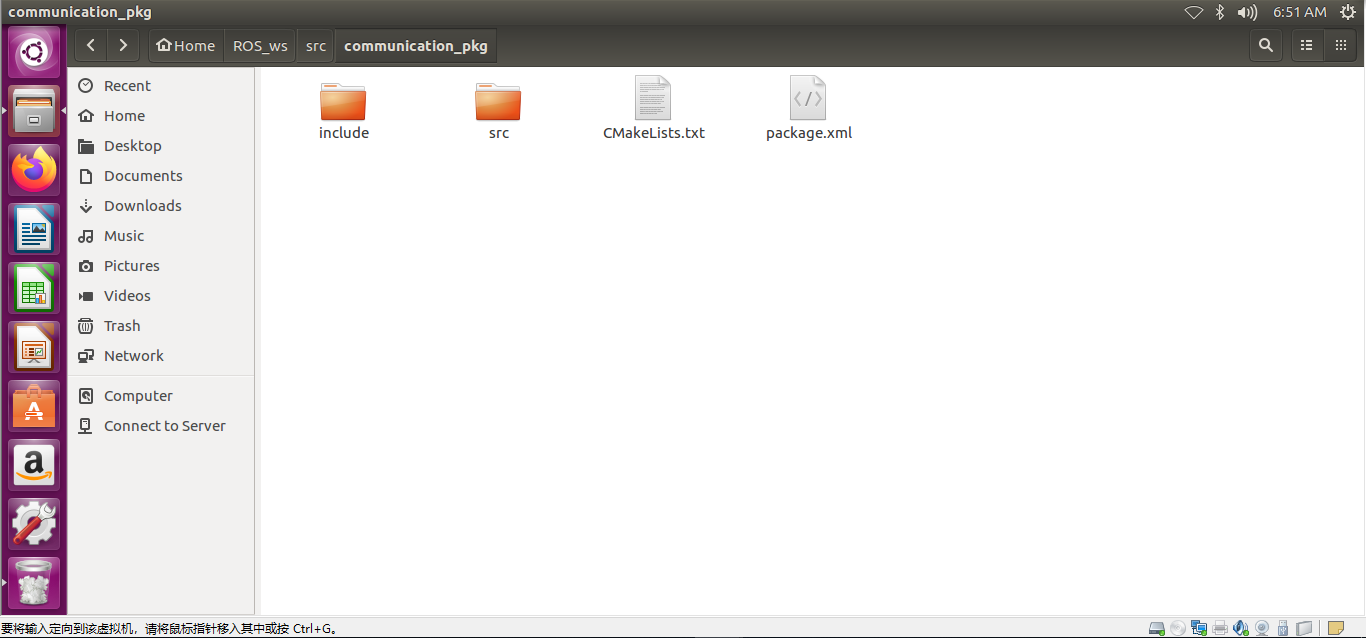
**ROS服务通信的编程实现**

**1.创建功能包**

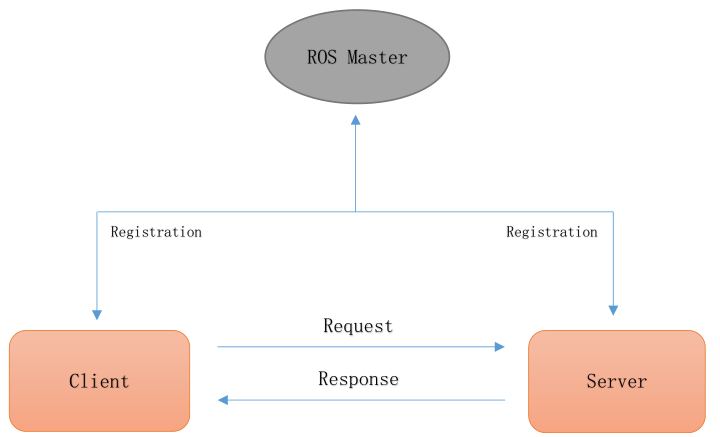
在ROS工作空间ROS\_ws的src文件夹目录下创建一个功能包，命名为communication\_pkg，并编译完成。



**2.节点编程与服务数据定义**

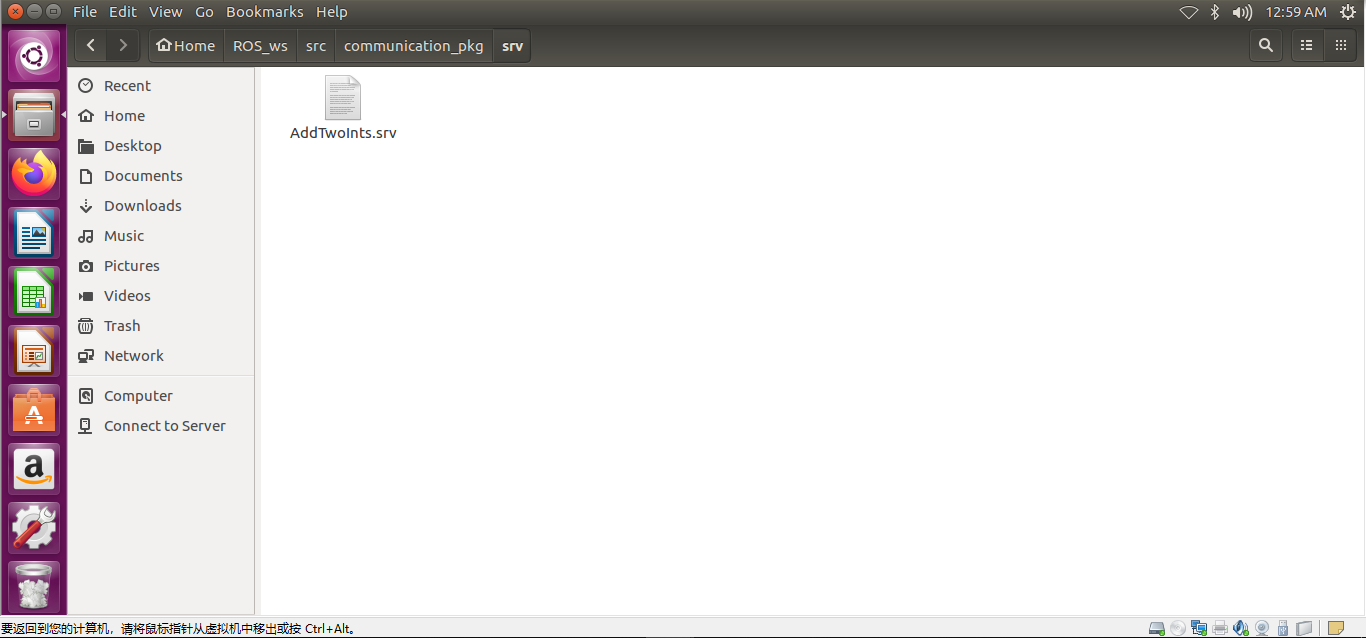
2.1案例说明

定义一个客户端Client，通过自定义的服务请求Request将两个整数发送给服务器Server，同时服务器Server将两个整数相加后，再通过自定义的服务应答Response将两个整数的和反馈到客户端Client。



2.2服务数据的定义

在功能包目录下创建一个新的文件夹，命名为srv，并在此文件夹中创建一个空文件AddTwoInts.srv。



在AddTwoInts.srv文件中输入以下代码，定义服务消息内容。

```

int64 a

int64 b

---

int64 sum

```

说明：

使用“---”作为分隔符，“---”上面代表服务请求的消息内容，“---”下面代表服务应答的消息内容。

相关资料：https://wiki.ros.org/srv

2.3创建.cpp文件

在功能包下面的src文件夹目录下创建一个空文件client.cpp。

2.4客户端编程

打开上面所创建的文件client.cpp，输入以下代码。

**```**

#include <cstdlib>

#include "ros/ros.h"

#include "communication\_pkg/AddTwoInts.h"

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

{

//初始化ROS节点

ros::init(argc, argv, "add\_two\_ints\_client");

//从终端获取数据

if(argc != 3)

{

ROS\_INFO("usage: add\_two\_ints\_client X Y");

return 1;

}

//创建句柄

ros::NodeHandle n;

//创建一个客户端，定义服务请求数据类型communication\_pkg::AddTwoInts

ros::ServiceClient client = n.serviceClient<communication\_pkg::AddTwoInts>("add\_two\_ints");

//定义服务消息内容

communication\_pkg::AddTwoInts srv;

srv.request.a = atoll(argv[1]);

srv.request.b = atoll(argv[2]);

//发布服务请求，等待应答结果

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;

}

```

说明：

头文件cstdlib.h是C++里面的一个常用函数库，等价于C中的stdlib.h，可以提供一些函数与符号常量。

头文件ros/ros.h包含了标准ROS类的声明，在每一个ROS程序中都需要包含它。

头文件communication\_pkg/AddTwoInts.h是由AddTwoInts.srv编译扩展得到，包含了针对C++类的定义等，它存放在工作空间的devel/include路径下的communication\_pkg文件夹中。

int main(int argc, char \*\*argv)是main函数的一种定义形式，**其参数argc和argv用于运行时,把命令行参数传入主程序，其中arg是指arguments，即参数。**

**(1)int argc：英文名为arguments count（参数计数），表示运行程序传送给main函数的命令行参数总个数,包括可执行程序名,其中当argc=1时表示只有一个程序名称,此时存储在argv[0]中。**

(2) **char \*\*argv：英文名为arguments value/vector（参数值），用来存放指向字符串参数的指针数组，每个元素指向一个参数，空格分隔参数，其长度为argc。数组下标从0开始，argv[argc]=NULL。argv[0]指向程序运行时的全路名；argv[1]指向程序在DOS命令中执行程序名后的第一个字符串；argv[2]指向执行程序名后的第二个字符串。**

ros::init(argc, argv, "add\_two\_ints\_client")的作用是初始化ROS节点，第三个参数表示节点名称，这个节点名是唯一的。

if(argc != 3){…}表示当终端命令行输入的参数数量不等于3，则执行这段代码。

ros::NodeHandle n;的作用是创建句柄，启动ROS节点。

ros::ServiceClient client = n.serviceClient<communication\_pkg::AddTwoInts>("add\_two\_ints");这是用户调用ROS服务的其中一种方法。它创建了一个客户端，其返回的ros::ServiceClient类的对象client之后将用于向ROS网络中名为add\_two\_ints的节点发送服务请求，服务请求的数据类型为communication\_pkg::AddTwoInts。

```

communication\_pkg::AddTwoInts srv;

srv.request.a = atoll(argv[1]);

srv.request.b = atoll(argv[2]);

```

这是创建服务请求和服务应答的其中一种方法。它定义了一个service\_example::AddTwoInts类型的对象，该对象中的成员正是srv文件中定义的a、b、sum，将终端命令行输入的两个数填充到数据成员a、b中。atoll()函数的作用是将字符串转换成长长整型数(long long int)，即srv.request.a = atoll(argv[1])是将命令行输入的字符串中的第一个数转换成长长整型数，并赋值给成员a，srv.request.b = atoll(argv[2])则同理。

if(client.call(srv)){…}表示向ROS网络发起服务请求。由于服务请求处于阻塞状态，需要等待请求发起完成后才能返回值。如果服务调用成功，则call()将返回true，并且srv.response中的值将有效。如果调用失败，则call()将返回false，并且srv.response中的值将无效。

ROS\_INFO()和ROS\_ERROR()用于输出日志信息，日志消息分为五个不同的严重级别，按照严重性程度递增，分别为DEBUG、INFO、WARN、ERROR、FATAL。

综上所述，实现一个客户端的步骤大致可分为以下几点：

(1)初始化ROS节点；

(2)创建client实例；

(3)发布服务请求数据；

(4)等待Server处理之后的应答结果。

2.5服务器编程

在src文件夹下再创建一个空文件server.cpp，输入以下代码。

```

#include "ros/ros.h"

#include "communication\_pkg/AddTwoInts.h"

//服务回调函数，输入为服务请求Req，输出为服务应答Res

bool add(communication\_pkg::AddTwoInts::Request &req,

communication\_pkg::AddTwoInts::Response &res)

{

//将两个数相加，并将结果存放在变量sum中

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)

{

//初始化ROS节点

ros::init(argc, argv, "add\_two\_ints\_server");

//创建句柄

ros::NodeHandle n;

//创建服务器，登记回调函数

ros::ServiceServer service = n.advertiseService("add\_two\_ints", add);

//循环等待

ROS\_INFO("Ready to add two ints.");

ros::spin();

return 0;

}

```

说明：

main函数中一开始都是类似的，初始化ROS节点，创建句柄，从而启动ROS节点。

ros::ServiceServer service = n.advertiseService("add\_two\_ints", add);的作用是创建服务，并将服务加入到ROS网络中。这个服务在ROS网络中以add\_two\_ints命名，并且作为唯一标识，以便于其他节点通过服务名称进行请求。当接收到服务请求后，则调用add()回调函数。

ros::spin()的作用是让程序进入自循环的挂起状态，从而让程序以最好的效率接收客户端的请求并调用回调函数。

服务回调函数的表示形式如下：

```

bool callback(MReq &request, MRes &response)

```

其中MReq和MRes与提供给advertiseService()的请求/应答的数据类型相匹配。回调函数的返回值若为true，则表示服务请求成功，并且服务应答已填充必要的数据。若返回值为false则表示服务请求失败，并且服务应答将不会发送给客户端。

回调函数bool add(communication\_pkg::AddTwoInts::Request &req, communication\_pkg::AddTwoInts::Response &res)的作用是实现两个int型整数求和的服务，两个整数从request获取，求和结果填充到response里，request与response的具体数据类型在srv文件中被定义。

综上所述，实现一个服务器的步骤大致可分为以下几点：

(1)初始化ROS节点；

(2)创建Server实例；

(3)循环等待服务请求，进入回调函数；

(4)在回调函数中完成服务功能的处理，并反馈应答数据。

附相关资料：

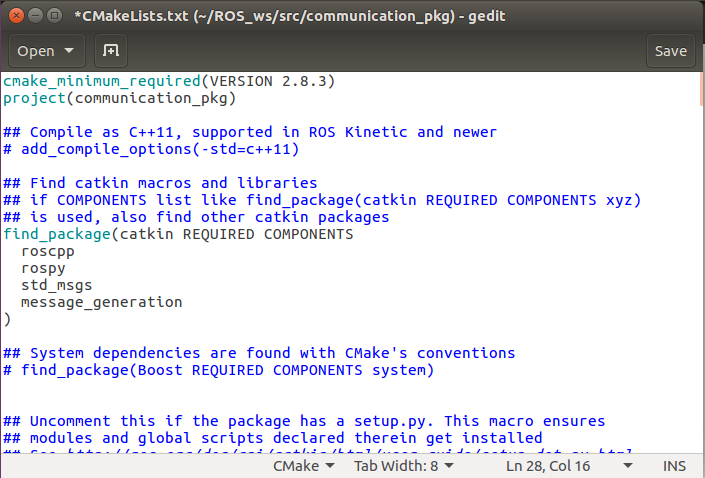
1. <http://wiki.ros.org/ROS/Tutorials/WritingServiceClient%28c%2B%2B%29>

2. <http://wiki.ros.org/roscpp/Overview/Services>

**3.配置与编译**

3.1在CMaKeLists.txt中添加编译选项

打开功能包中的CMaKeLists.txt文件，在如下位置的find\_package中添加功能包，以便于（节点）调用它们生成消息。

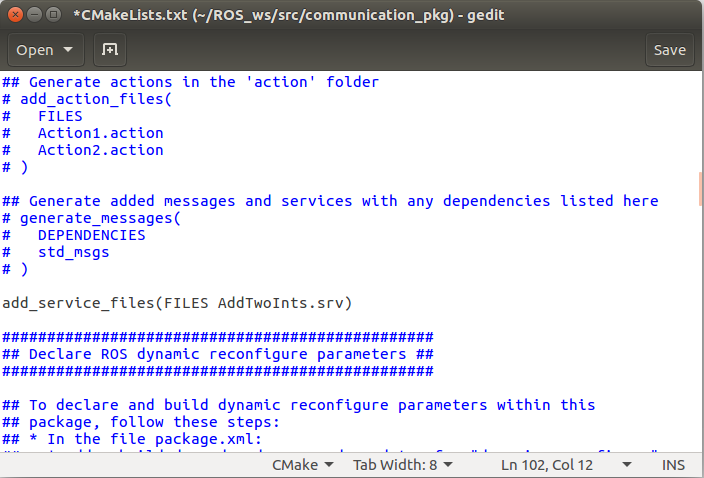


在如下位置添加相关的.srv文件，确保了CMake在重新配置时知道这些新添加的.srv文件，同时添加.srv文件在生成消息时的所有依赖项（功能包）。

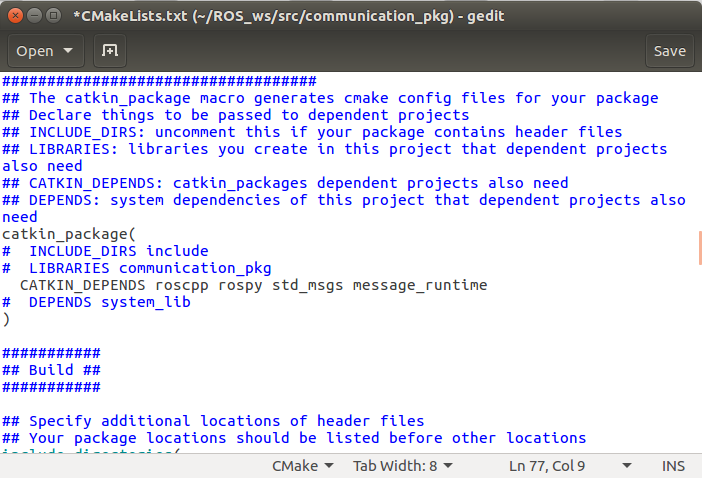
```

add\_service\_files(FILES AddTwoInts.srv)

```



将如下位置中CATLIN\_DEPENDS前面的“#”去掉。



在如下位置进行配置，add\_executable(client src/client.cpp)的作用是将src文件夹下的client.cpp文件编译成名为client的可执行文件。target\_link\_libraries(client ${catkin\_LIBRARIES})的作用是将client可执行文件与ROS相关的库链接。add\_dependencies(client ${PROJECT\_NAME}\_gencpp)的作用是将client可执行文件与一些动态生成的文件链接。

```

add\_executable(server src/server.cpp)

target\_link\_libraries(server ${catkin\_LIBRARIES})

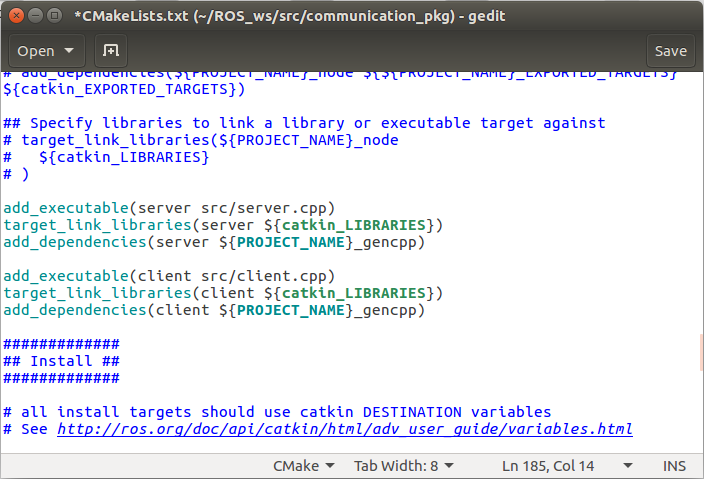
add\_dependencies(server ${PROJECT\_NAME}\_gencpp)

add\_executable(client src/client.cpp)

target\_link\_libraries(client ${catkin\_LIBRARIES})

add\_dependencies(client ${PROJECT\_NAME}\_gencpp)

```



3.2在package.xml中添加功能包依赖

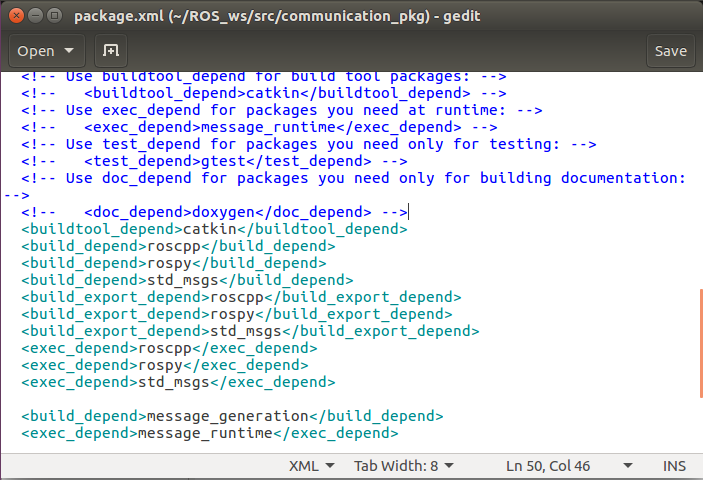
打开功能包中的package.xml文件，在如下位置添加功能包依赖。<buildtool\_depend>catkin</buildtool\_depend> 表示需要依赖catkin工具包用于使用catkin\_make指令进行编译。 <build\_export\_depend>roscpp</build\_export\_depend>表示编译后导出相关文件需要依赖roscpp功能包。<build\_depend>message\_generation</build\_depend>表示在编译时会依赖一个动态产生message的功能包。<exec\_depend>message\_runtime</exec\_depend>表示在运行时会依赖message\_runtime的功能包。

```

<build\_depend>message\_generation</build\_depend>

<exec\_depend>message\_runtime</exec\_depend>

```



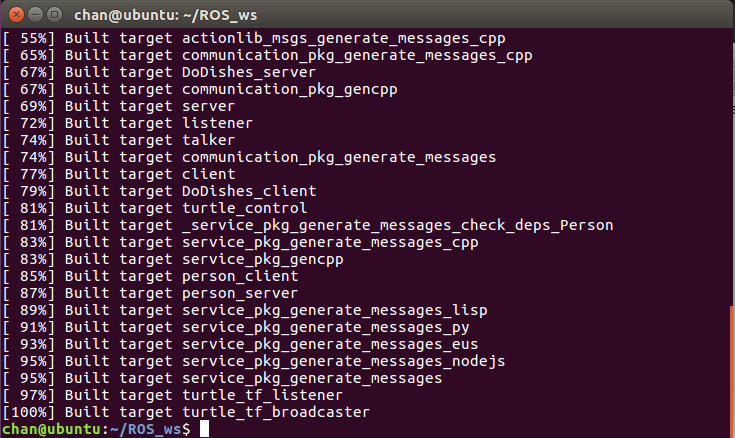
3.3编译文件

在/ROS\_ws文件夹路径下打开一个新的终端，输入以下代码进行编译。

```

$ catkin\_make

```



编译完成后，输入以下代码运行主节点。

```

$ roscore

```

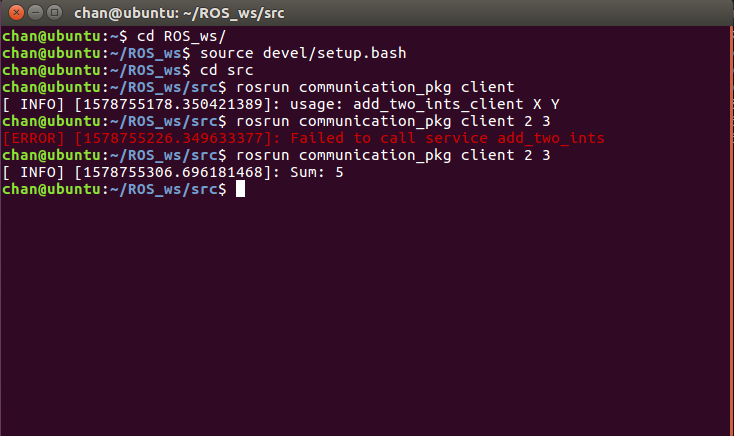


打开一个新的终端，配置环境变量后，输入以下代码运行节点。

```

$ rosrun communication\_pkg client 整数1 整数2

```



如果rosrun指令后面缺少需要相加的两个数，则会报如下错误。

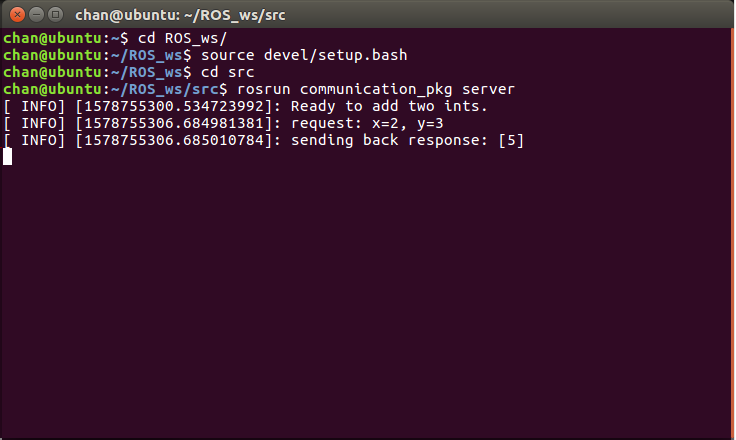


打开一个新的终端，配置环境变量后，输入以下代码运行节点。

```

$ rosrun communication\_pkg server

```



若想停止运行，关闭终端，使用快捷键Ctrl+c即可。