发布者

```
int main(int argc, char *argv[])
```

程序的入口。argc 和 argv 是命令行参数。

```
ros::init(argc, argv, "send_pkg");
```

初始化ROS系统，设置节点名称为"send_pkg"。

```
ros::NodeHandle nh;
```

创建一个节点，用于与ROS系统交互。

```
ros::Publisher pub = nh.advertise<std_msgs::Int64>("number", 10);
```

创建一个发布者，发布类型为std_msgs::Int64的消息到名为"number"的话题，话题队列大小为10。

```
std_msgs::Int64 msg;
```

定义一个std_msgs::Int64类型的消息变量msg。

```
int64_t number1 = 256;
int64_t number2 = 512;
int64_t number3 = 1024;
```

定义三个int64_t类型的变量，分别赋值为256、512、1024。

```
sleep(1);
msg.data = number1;
pub.publish(msg);
sleep(1);
msg.data = number2;
pub.publish(msg);
sleep(1);
msg.data = number3;
pub.publish(msg);
```

暂停一秒等待订阅者连接。
发布三个消息，分别赋值为number1、number2、number3。

```
ros::spinOnce();
```

处理所有待处理的ROS事件。

```
return 0;
```

订阅者

```
int num=0;
```

定义一个全局整数变量num，并初始化为0。

```
void chatterCallback(const std_msgs::Int64::ConstPtr& msg)
```

定义一个回调函数chatterCallback。

```
{
    //ROS_WARN("I heard: [%ld]", msg->data);
    switch(num%3)
    {
        case 0:
            printf("X: %ld   ", msg->data);
            break;
        case 1:
            printf("Y: %ld   ", msg->data);
            break;
        case 2:
            printf("Z: %ld   ", msg->data);
            break;
        default:
            break;
    }
    num++;
    fflush(stdout);//刷新缓冲区，立即输出
}
```

回调函数函数体，打印接收到的消息，并将num自增1。

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

程序入口。argc 和 argv 是命令行参数。

```
ros::init(argc, argv, "accept_node");
```

初始化ROS系统，设置节点名称为"accept_node"。

```
ros::NodeHandle nh;
```

创建一个节点，用于与ROS系统交互。

```
ros::Subscriber sub = nh.subscribe("number", 10, chatterCallback);
```

创建一个订阅者，订阅名为"number"的话题，队列大小为10，当接收到消息时调用chatterCallback函数。

```
while(ros::ok())
```

当ROS系统正常运行时，执行循环。

```
{
ros::spin();
}
```

进入一个循环，处理所有ROS事件，包括回调函数的调用。

```
return 0;
```

返回0，表示程序正常结束。