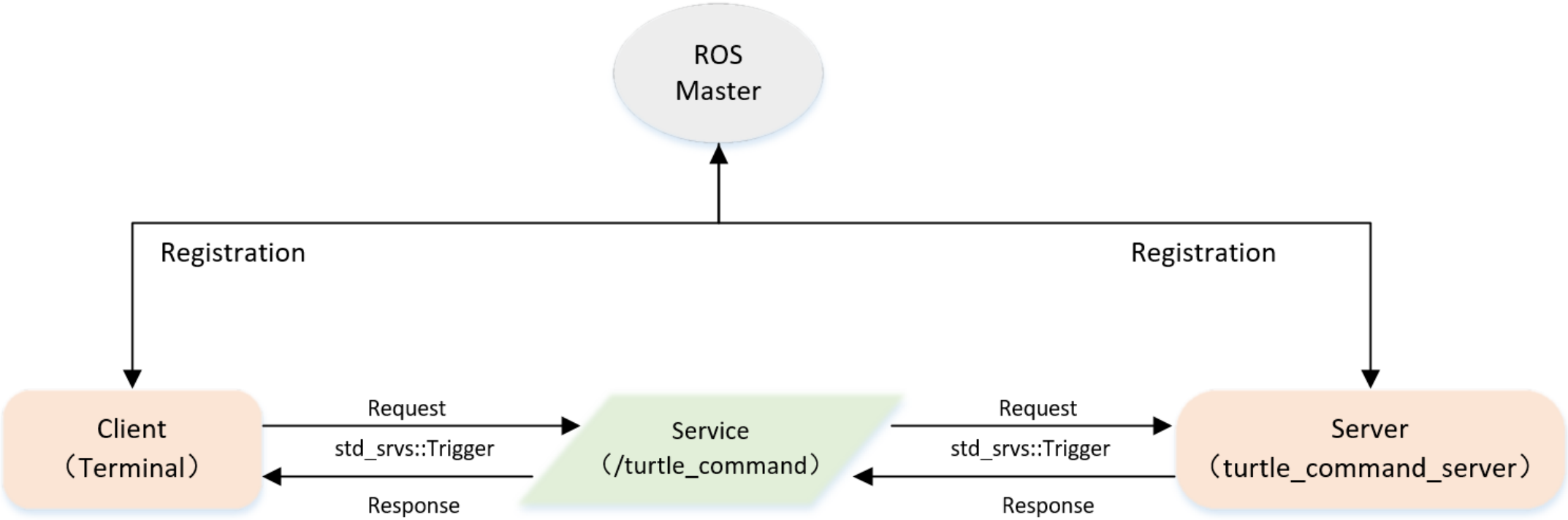
古月居 GYH.AI ROS

ROS入门
21讲

14.服务端Server的编程实现

主讲人：古月

服务模型



服务模型（客户端/服务器）

创建服务器代码（C++）

```cpp
int main(int argc, char **argv)
{
    // ROS节点初始化
    ros::init(argc, argv, "turtle_command_server");

    // 创建节点句柄
    ros::NodeHandle n;

    // 创建一个名为/turtle_command的server，注册回调函数commandCallback
    ros::ServiceServer command_service = n.advertiseService("/turtle_command", commandCallback);

    // 创建一个Publisher，发布名为/turtle1/cmd_vel的topic，消息类型为geometry_msgs::Twist，队列长度10
    turtle_vel_pub = n.advertise<geometry_msgs::Twist>("/turtle1/cmd_vel", 10);

    // 循环等待回调函数
    ROS_INFO("Ready to receive turtle command.");

    // 设置循环的频率
    ros::Rate loop_rate(10);

    while(ros::ok())
    {
        // 查看一次回调函数队列
        ros::spinOnce();

        // 如果标志为true，则发布速度指令
        if(pubCommand)
        {
            geometry_msgs::Twist vel_msg;
            vel_msg.linear.x = 0.5;
            vel_msg.angular.z = 0.2;
            turtle_vel_pub.publish(vel_msg);
        }

        //按照循环频率延时
        loop_rate.sleep();
    }

    return 0;
}
```

turtle_command_server.cpp

```cpp
/**
 * 该例程将执行/turtle_command服务，服务数据类型std_srvs/Trigger
 */

#include <ros/ros.h>
#include <geometry_msgs/Twist.h>
#include <std_srvs/Trigger.h>

ros::Publisher turtle_vel_pub;
bool pubCommand = false;

// service回调函数，输入参数req，输出参数res
bool commandCallback(std_srvs::Trigger::Request  &req,
                     std_srvs::Trigger::Response &res)
{
    pubCommand = !pubCommand;

    // 显示请求数据
    ROS_INFO("Publish turtle velocity command [%s]", pubCommand==true?"Yes":"No");

    // 设置反馈数据
    res.success = true;
    res.message = "Change turtle command state!"

    return true;
}
```

如何实现一个服务器

- 初始化ROS节点；
- 创建Server实例；
- 循环等待服务请求，进入回调函数；
- 在回调函数中完成服务功能的处理，并反馈应答数据。

配置服务器代码编译规则

```
## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/learning_service_node.cpp)

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

add_executable(turtle_spawn src/turtle_spawn.cpp)
target_link_libraries(turtle_spawn ${catkin_LIBRARIES})

add_executable(turtle_command_server src/turtle_command_server.cpp)
target_link_libraries(turtle_command_server ${catkin_LIBRARIES})
```

如何配置CMakeLists.txt中的编译规则

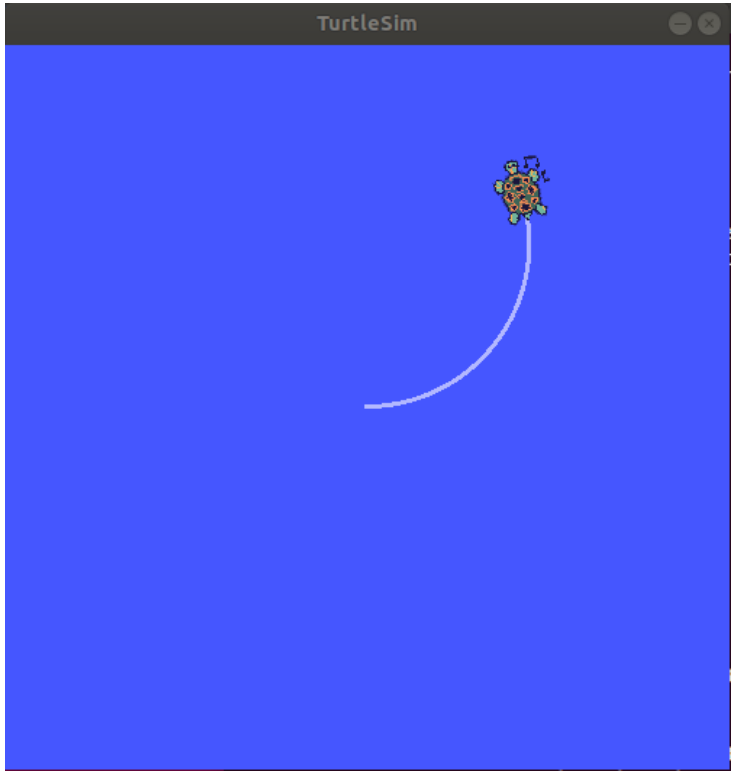
- 设置需要编译的代码和生成的可执行文件；
- 设置链接库；

```
add_executable(turtle_command_server src/turtle_command_server.cpp)
target_link_libraries(turtle_command_server ${catkin_LIBRARIES})
```

CMakeLists.txt

编译并运行服务器

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun turtlesim turtlesim_node
$ rosrun learning_service turtle_command_server
$ rosservice call /turtle_command "{}"
```




```
hcx@hcx-vpc:~/catkin_ws$ rosrun learning_service turtle_command_server
[ INFO] [1562230799.802379365]: Ready to receive turtle command.
[ INFO] [1562230816.003741112]: Publish turtle velocity command [Yes]
[ INFO] [1562230821.203543112]: Publish turtle velocity command [No]
```

```
hcx@hcx-vpc:~$ rosservice call /turtle_command "{}"
success: True
message: "Change turtle command state!"
hcx@hcx-vpc:~$ rosservice call /turtle_command "{}"
success: True
message: "Change turtle command state!"
```

创建服务器代码（Python）

```python
pubCommand = False;
turtle_vel_pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)

def command_thread():
    while True:
        if pubCommand:
            vel_msg = Twist()
            vel_msg.linear.x = 0.5
            vel_msg.angular.z = 0.2
            turtle_vel_pub.publish(vel_msg)

        time.sleep(0.1)

def commandCallback(req):
    global pubCommand
    pubCommand = bool(1-pubCommand)

    # 显示请求数据
    rospy.loginfo("Publish turtle velocity command![%d]", pubCommand)

    # 反馈数据
    return TriggerResponse(1, "Change turtle command state!")

def turtle_command_server():
    # ROS节点初始化
    rospy.init_node('turtle_command_server')

    # 创建一个名为/turtle_command的server，注册回调函数commandCallback
    s = rospy.Service('/turtle_command', Trigger, commandCallback)

    # 循环等待回调函数
    print "Ready to receive turtle command."

    thread.start_new_thread(command_thread, ())
    rospy.spin()

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

turtle_command_server.py

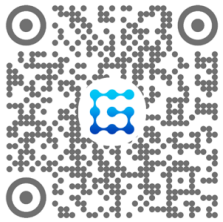
如何实现一个服务器

- 初始化ROS节点；
- 创建Server实例；
- 循环等待服务请求，进入回调函数；
- 在回调函数中完成服务功能的处理，并反馈应答数据。

感谢观看

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注



古月居

古月学院

古月居GYH