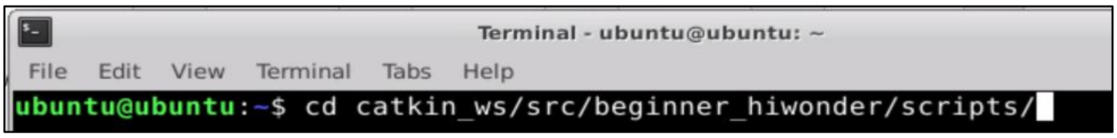
Lesson 8 Write A Simple Client

This section takes the creation of a simple service (Client) node turtle_spawn.py as an example to explain and this node publishes a request for the client to spawn a new turtle by means of a program.

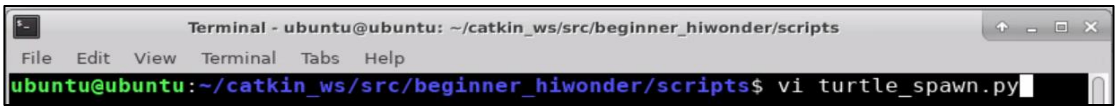
1. Configure Client Code Compilation Rule

1) Enter “cd catkin_ws/src/beginner_hiwonder/scripts/” command and press “Enter”.


Terminal - ubuntu@ubuntu: ~
File Edit View Terminal Tabs Help

```
ubuntu@ubuntu:~$ cd catkin_ws/src/beginner_hiwonder/scripts/
```

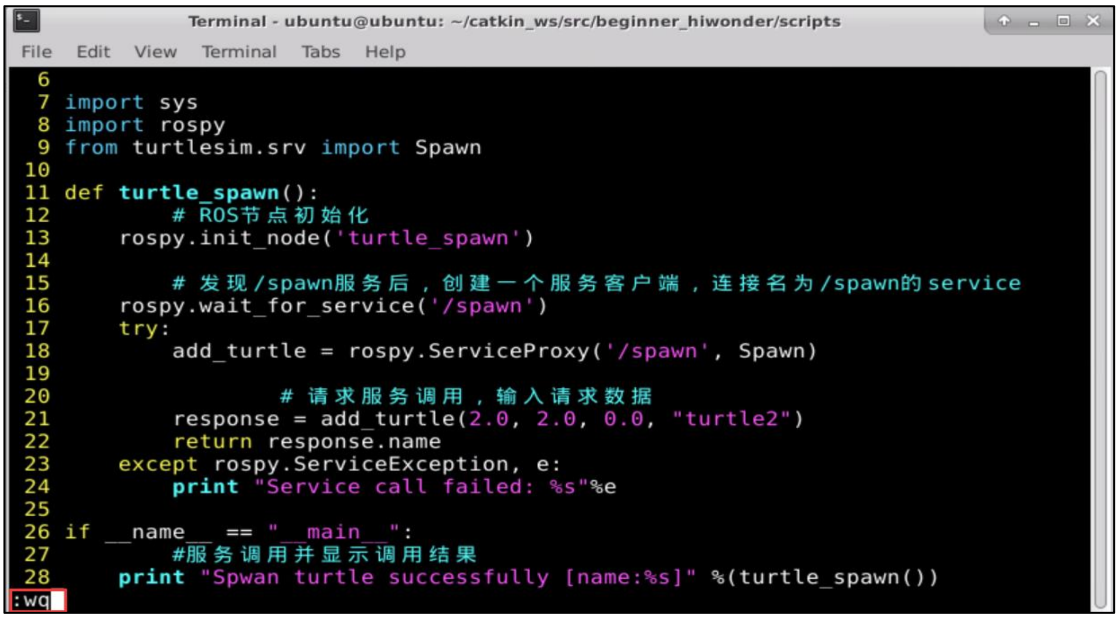
2) Enter “vi turtle_spawn.py” command to edit program and copy the following program. If want to program, you can press “i”, and then press “Esc” to enter “:wq” to exit and save.


Terminal - ubuntu@ubuntu: ~/catkin_ws/src/beginner_hiwonder/scripts
File Edit View Terminal Tabs Help

```
ubuntu@ubuntu:~/catkin_ws/src/beginner_hiwonder/scripts$ vi turtle_spawn.py
```

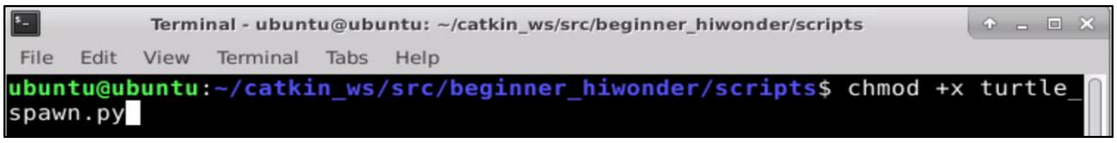
```
#!/usr/bin/env python

# -*- coding: utf-8 -*-




# This routine will request /spawn service and the service data type is turtlesim::Spawn



import sys

import rospy
```

```
from turtlesim.srv import Spawn


def turtle_spawn():
    # Initialize ROS node
    rospy.init_node('turtle_spawn')


    # After finding the /spawn service, create a service client, and then connect the
service named /spawn.
    rospy.wait_for_service('/spawn')
    try:
        add_turtle = rospy.ServiceProxy('/spawn', Spawn)


        # Request service call and input request data
        response = add_turtle(2.0, 2.0, 0.0, "turtle2")
        return response.name
    except rospy.ServiceException, e:
        print "Service call failed: %s"%e


if __name__ == "__main__":
    #The service calls and displays the result of call.
    print "Spwan turtle successfully [name:%s]" %(turtle_spawn())
```


Terminal - ubuntu@ubuntu: ~/catkin_ws/src/beginner_hiwonder/scripts
File Edit View Terminal Tabs Help

```

import sys
import rospy
from turtlesim.srv import Spawn

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

        # 发现/spawn服务后，创建一个服务客户端，连接名为/spawn的service
    rospy.wait_for_service('/spawn')
    try:
        add_turtle = rospy.ServiceProxy('/spawn', Spawn)

                # 请求服务调用，输入请求数据
        response = add_turtle(2.0, 2.0, 0.0, "turtle2")
        return response.name
    except rospy.ServiceException, e:
        print "Service call failed: %s"%e

if __name__ == "__main__":
        #服务调用并显示调用结果
    print "Spwan turtle successfully [name:%s]" %(turtle_spawn())
:wq
```

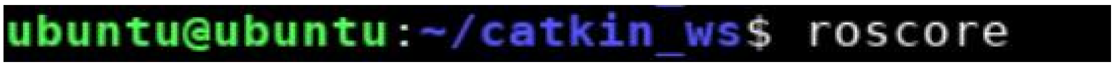
3) Enter “chmod +x turtle_spawn.py” command and press “Enter” to give the executable permission to the saved turtle_spawn.py.


Terminal - ubuntu@ubuntu: ~/catkin_ws/src/beginner_hiwonder/scripts
File Edit View Terminal Tabs Help

```
ubuntu@ubuntu:~/catkin_ws/src/beginner_hiwonder/scripts$ chmod +x turtle_
spawn.py
```

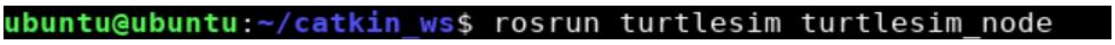
2. Run Client

1) Enter “roscore” command to start node manager.



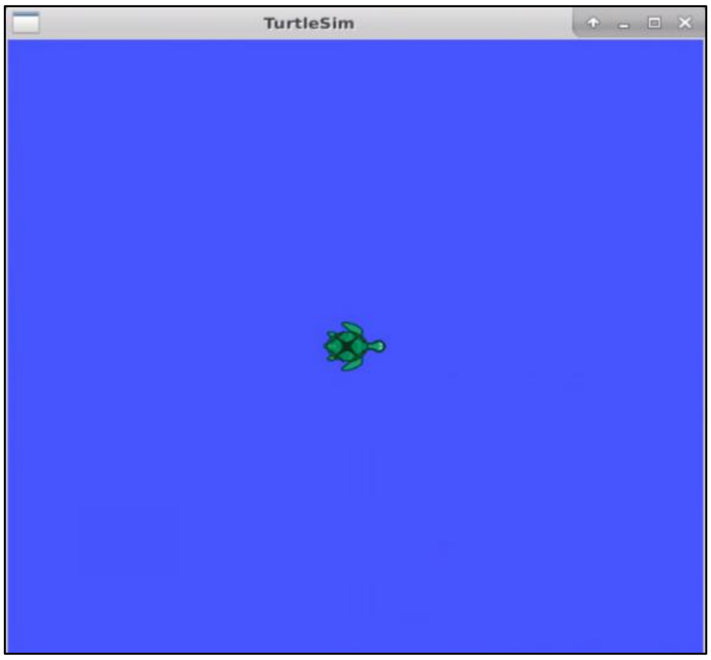
```
ubuntu@ubuntu:~/catkin_ws$ roscore
```

2) Enter “rosrun turtlesim turtlesim_node” command and press “enter” to run turtlesim.

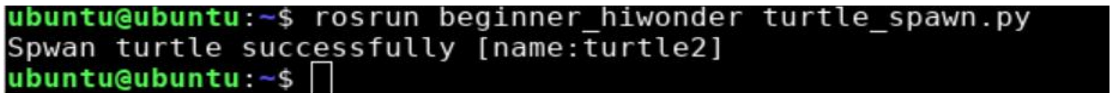


```
ubuntu@ubuntu:~/catkin_ws$ rosrun turtlesim turtlesim_node
```

At this time, the interface will pop up the turtlesim window, as the figure shown below:



3) Open a new terminal. Enter “rosrun beginner_hiwonder turtle_spawn.py” command and press “Enter” to run the client.



```
ubuntu@ubuntu:~$ rosrun beginner_hiwonder turtle_spawn.py
Spwan turtle successfully [name:turtle2]
ubuntu@ubuntu:~$ 
```

At this time, client will send the request to server and respond to start another turtle.