

## 5. ROS2launch file startup

---

In ROS2, launch is used for multi-node startup and configuring program runtime parameters, etc. The launch file formats in ROS2 are xml, yaml and python. This lesson takes the python format launch file as an example. Compared with the other two formats, the python format is more flexible:

- python has numerous libraries of functions that can be used in the launch file;
- ROS2 generic launch features and specific launch features are written in Python, thus providing access to launch features that may not be publicly available in XML and YAML;

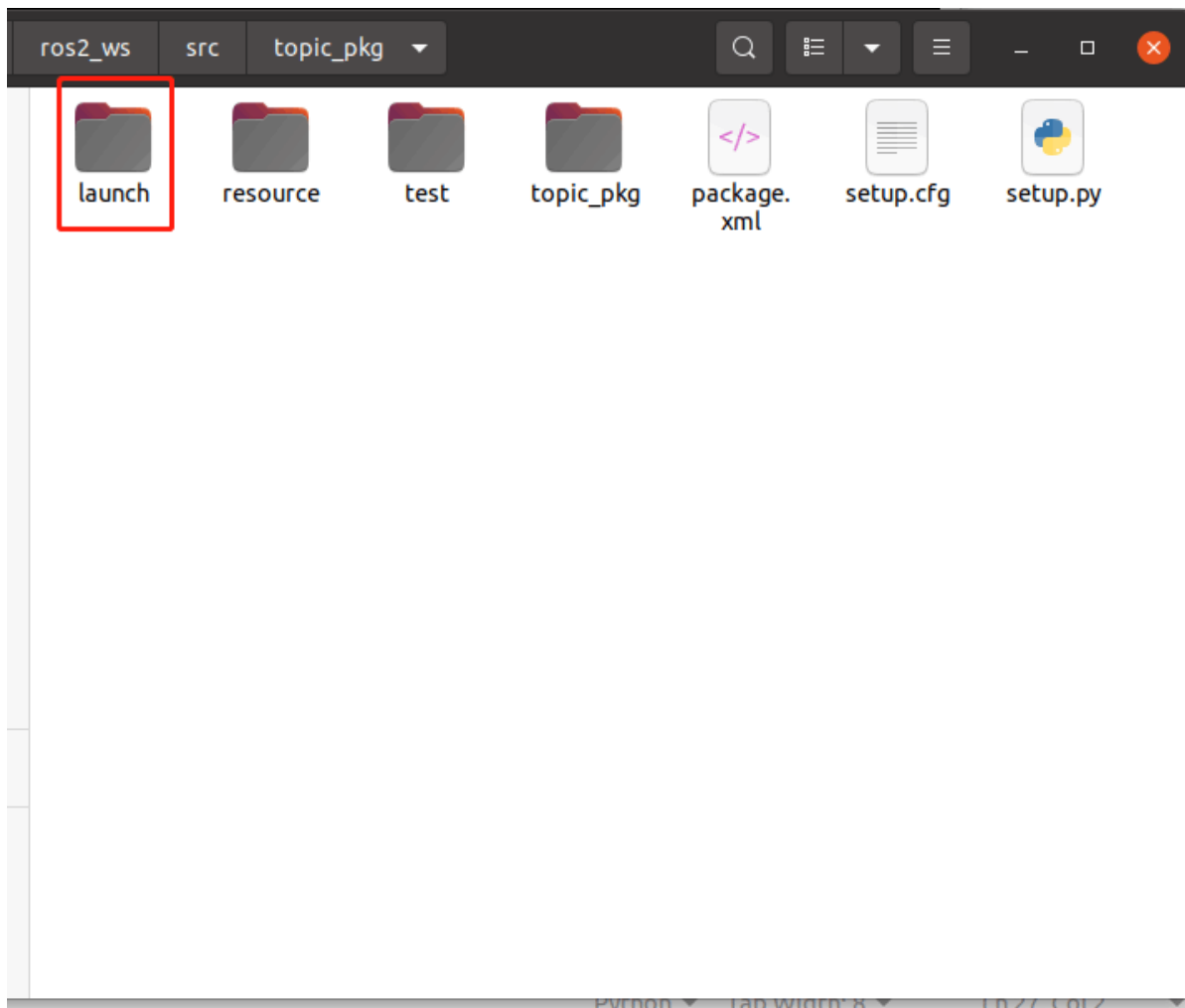
The main thing about writing a ROS2 launch file using the python language is to abstract each node, file, script, etc., into an action that is launched using a unified interface, with the main structure being that the

```
def generate_launch_description():  
    return LaunchDescription([  
        action_1,  
        action_2,  
        ...  
        action_n  
    ])
```

### 1. Create a new folder to store the launch file.

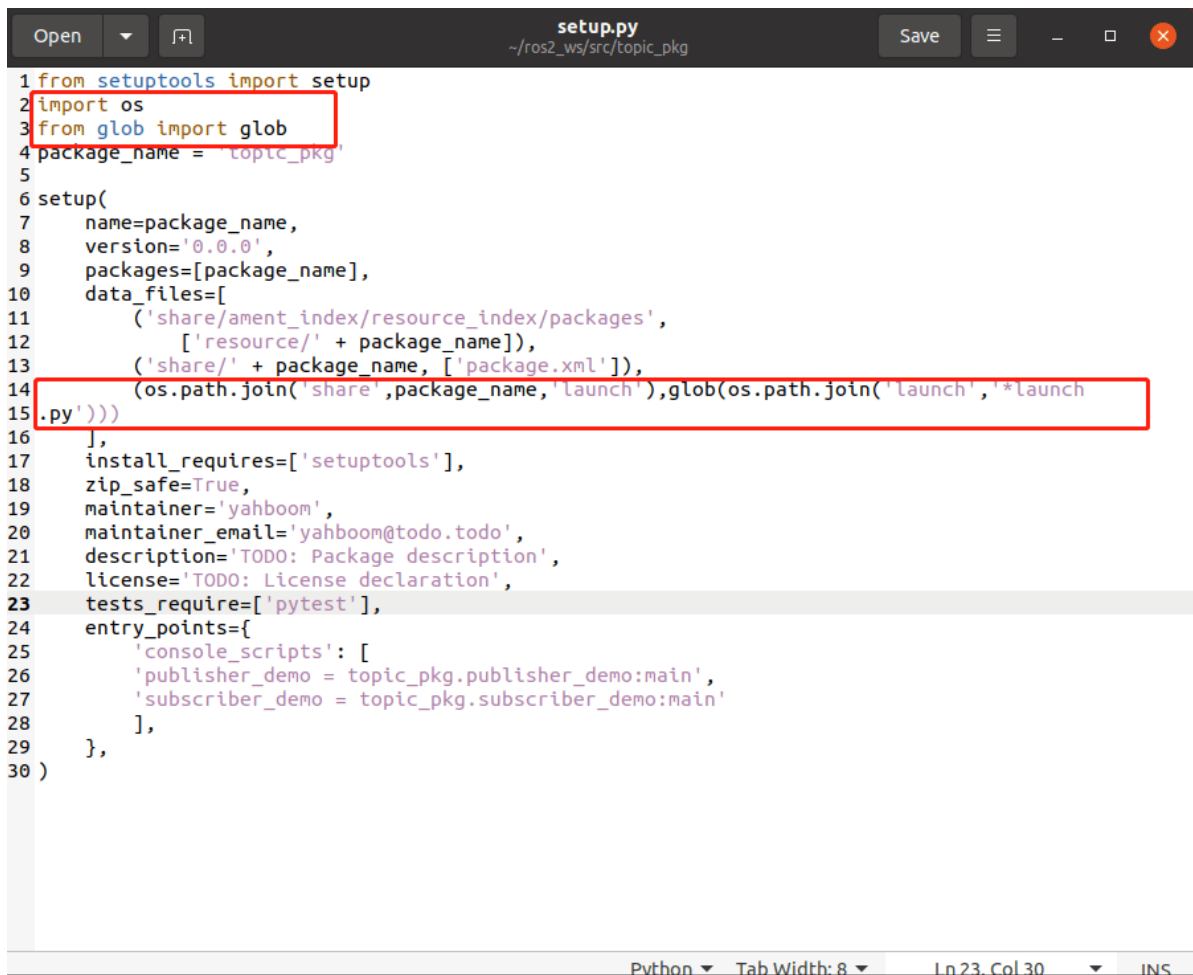
We create a new folder to store the launch file under the path of the function package we created earlier, and type in the terminal.

```
cd ~/ros2_ws/src/topic_pkg  
mkdir launch
```



The launch file is often named `LaunchName_launch.py`, where `LaunchName` is customized and `_launch.py` is often considered fixed. You need to modify the `setup.py` file under the function `package` to add the files under the launch path in order to compile the `.py` file for execution.

```
#1、导入相关的头文件 #1. Importing relevant header files
import os
from glob import glob
#2、在data_files的列表中，加上launch路径以及路径下的launch.py文件
#2. In the list of data_files, add the path of launch and the launch.py file
under the path.
(os.path.join('share', package_name, 'launch'), glob(os.path.join('launch', '*launch
.py'))))
```



```
1 from setuptools import setup
2 import os
3 from glob import glob
4 package_name = 'topic_pkg'
5
6 setup(
7     name=package_name,
8     version='0.0.0',
9     packages=[package_name],
10    data_files=[
11        ('share/ament_index/resource_index/packages',
12         ['resource/' + package_name]),
13        ('share/' + package_name, ['package.xml']),
14        (os.path.join('share', package_name, 'launch'), glob(os.path.join('launch', '*launch
15        .py'))),
16    ],
17    install_requires=['setuptools'],
18    zip_safe=True,
19    maintainer='yahboom',
20    maintainer_email='yahboom@todo.todo',
21    description='TODO: Package description',
22    license='TODO: License declaration',
23    tests_require=['pytest'],
24    entry_points={
25        'console_scripts': [
26            'publisher_demo = topic_pkg.publisher_demo:main',
27            'subscriber_demo = topic_pkg.subscriber_demo:main'
28        ],
29    },
30 )
```

## 2. Write a single Node node launch

Terminal input.

```
cd ~/ros2_ws/src/topic_pkg/launch
gedit single_node_launch.py
```

Copy the following into that file.

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    turtle_node = Node(
        package='turtlesim',
        executable='turtlesim_node',
    )

    launch_description = LaunchDescription([turtle_node])
    return launch_description
```

### 2.1, Compile Workspace

Terminal input, the

```
cd ~/ros2_ws
colcon build
```

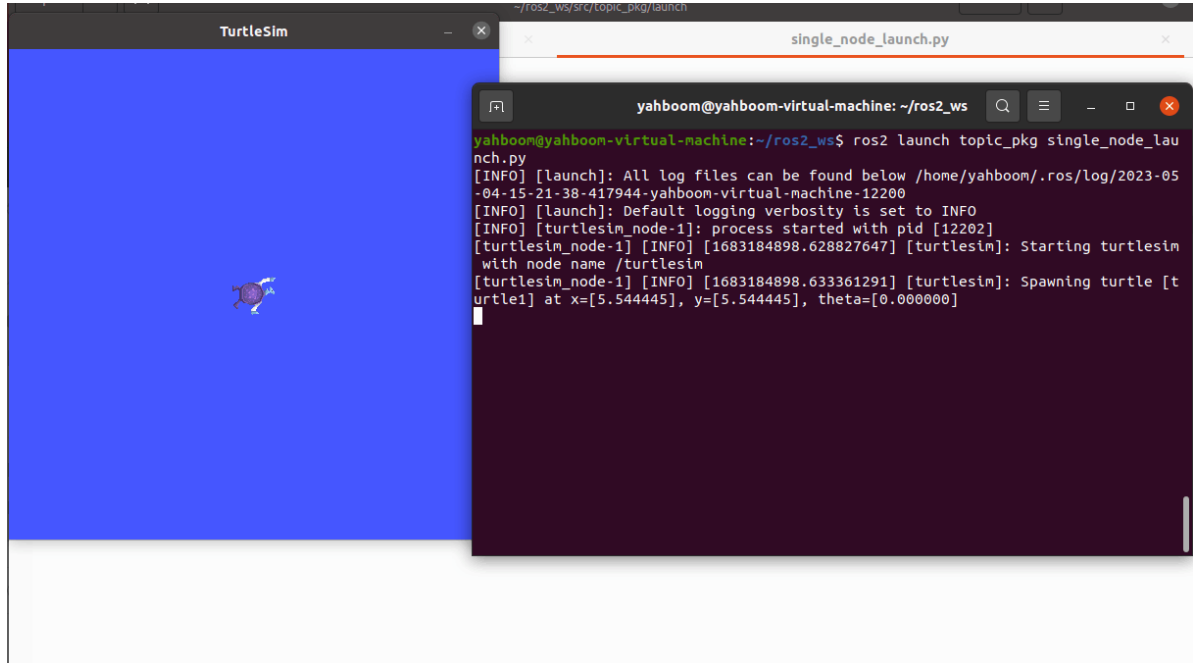
After the compilation is complete, refresh the environment variables in the workspace.

```
source ~/ros2_ws/install/setup.bash
```

## 2.2. Run the program

Terminal input,

```
ros2 launch topic_pkg single_node_launch.py
```



After the program is run, it will be running the node of the little turtle.

## 2.3, Source Code Analysis

### 1), Import related libraries

```
from launch import LaunchDescription
from launch_ros.actions import Node
```

### 2), Define a function generate\_launch\_description, and return a launch\_description

```
def generate_launch_description():
    turtle_node = Node(
        package='turtlesim',
        executable='turtlesim_node',
    )
    launch_description = LaunchDescription([turtle_node])
    return launch_description
```

A variable `turtle_node` is defined as the return value of a node startup that calls the `Node` function, starting the two important parameters, `package` and `executable`.

- `package`: indicates the function package, represents the name of the function package.
- `executable`: represents the program to be executed, the name of the executable program.

Then define a variable `launch_description` as the return value after calling the `LaunchDescription` function, if there are more than one node to start, just add it at the end.

```
launch_description = LaunchDescription([turtle_node])
```

Finally, return launch\_description.

### 3. Write launch for multiple Node nodes

Terminal input.

```
cd ~/ros2_ws/src/topic_pkg/launch
gedit multi_node_launch.py
```

Copy the following into that file.

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    pub_node = Node(
        package='topic_pkg',
        executable='publisher_demo',
        output='screen'
    )
    sub_node = Node(
        package='topic_pkg',
        executable='subscriber_demo',
        output='screen'
    )
    launch_description = LaunchDescription([pub_node, sub_node])
    return launch_description
```

#### 3.1, Compile Workspace

Terminal input,

```
cd ~/ros2_ws
colcon build
```

After the compilation is complete, refresh the environment variables in the workspace.

```
source ~/ros2_ws/install/setup.bash
```

#### 3.2. Run the program

Terminal input,

```
ros2 launch topic_pkg multi_node_launch.py
```

If the terminal doesn't print anything, we can check to see which nodes are up to verify that they're up, by typing in the terminal.

```
ros2 node list
```

```
yahboom@yahboom-virtual-machine:~$ ros2 node list
/publisher_node
/subscriber_node
```

As can be seen from the above figure, two nodes are started, which just correspond to the two programs in the launch file.

### 3.3. Source Code Analysis

Roughly the same as `simple_node_launch.py`, but with one extra node, and in `launch_description` = `LaunchDescription([pub_node,sub_node])`.

## 4. Topic name mapping in the launch file

Terminal input.

```
cd ~/ros2_ws/src/topic_pkg/launch
gedit remap_name_launch.py
```

Copy the following into that file.

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    turtle_node = Node(
        package='turtlesim',
        executable='turtlesim_node',
        remappings=[("/turtle1/cmd_vel", "/cmd_vel")]
    )
    launch_description = LaunchDescription([turtle_node])
    return launch_description
```

### 4.1, Compile Workspace

Terminal input,

```
cd ~/ros2_ws
colcon build
```

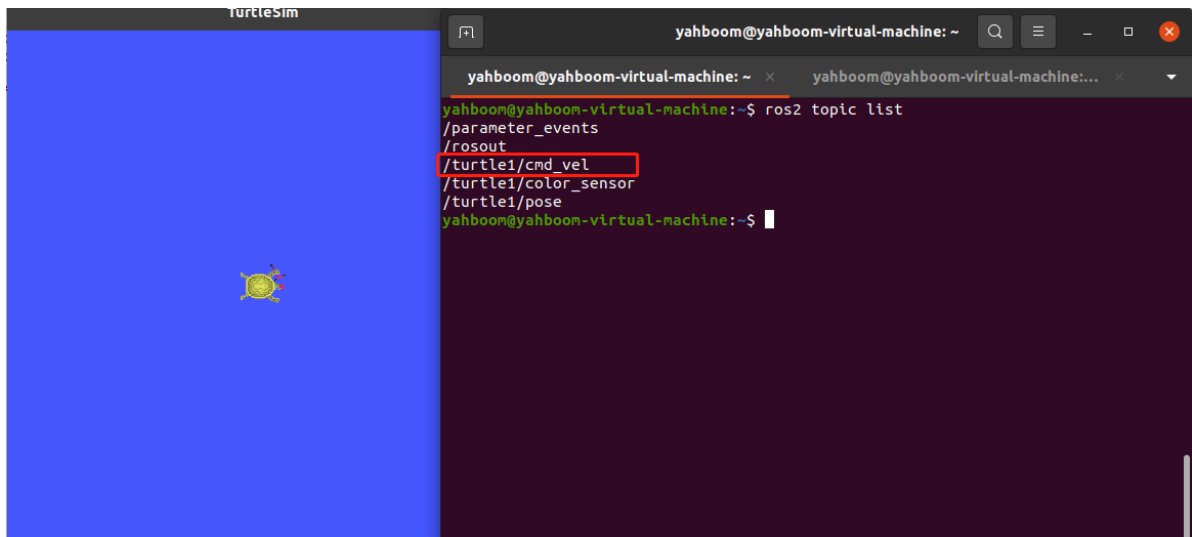
After the compilation is complete, refresh the environment variables in the workspace.

```
source ~/ros2_ws/install/setup.bash
```

### 4.2. Run the program

Let's see what the name of the speed topic for the little turtles was before we remapped the topic by typing, in the terminal.

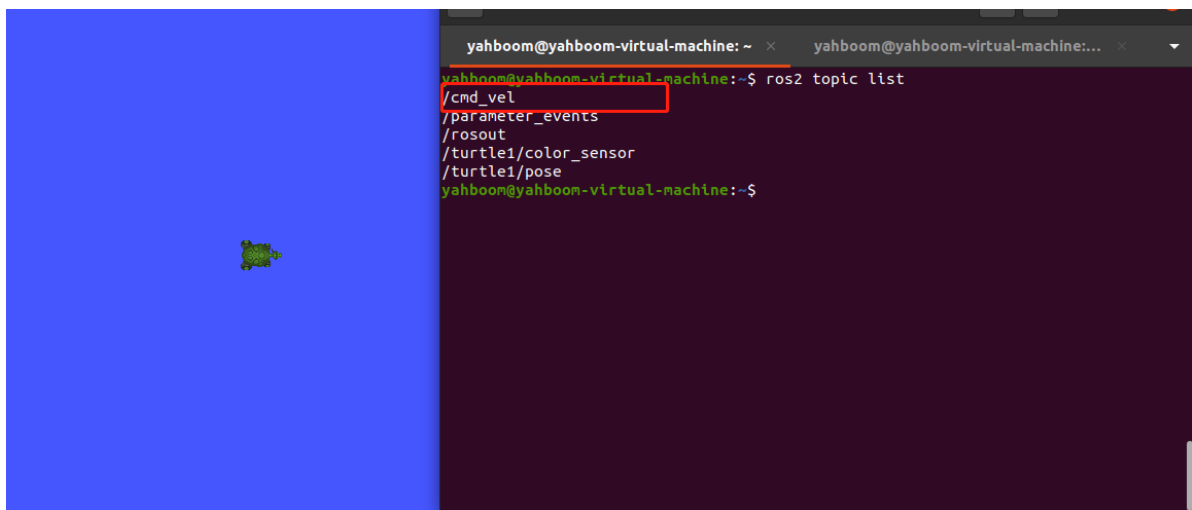
```
ros2 launch topic_pkg single_node_launch.py
ros2 topic list
```



The topic here is /turtle1/cmd\_vel.

Run the program again after remapping the topic to see what the name of the velocity topic is that Turtle subscribes to, and the terminal enters.

```
ros2 launch topic_pkg remap_name_launch.py
ros2 topic list
```



As can be seen from the above figure, the velocity topic name is remapped and the mapped name of the Little Turtle velocity topic is /cmd\_vel.

### 4.3. Source Code Analysis

Modifications were made to the original single\_node\_launch.py, mainly by adding the following parts.

```
remappings=[("/turtle1/cmd_vel", "/cmd_vel")]
```

Here we are remapping the original /turtle1/cmd\_vel to /cmd\_vel, with the original topic name in the front and the topic name we want to change to in the back.

## 5. Launch File Launching the Launch File

Terminal input.

```
cd ~/ros2_ws/src/topic_pkg/launch
gedit include_launch.py
```

Copy the following into that file.

```
import os
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource

def generate_launch_description():
    node1 = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
            get_package_share_directory('topic_pkg'), 'launch'),
            '/multi_node_launch.py'])
    )
    node2 = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
            get_package_share_directory('topic_pkg'), 'launch'),
            '/single_node_launch.py'])
    )
    return LaunchDescription([node1, node2])
```

### 5.1, Compile Workspace

Terminal input,

```
cd ~/ros2_ws
colcon build
```

After the compilation is complete, refresh the environment variables in the workspace.

```
source ~/ros2_ws/install/setup.bash
```

### 5.2. Run the program

Terminal input,

```
ros2 launch topic_pkg include_launch.py
```

This launch file will contain the launch of two launch files, `simple_node_launch.py` and `multi_node_launch.py`. You can check to see if the nodes of these launch files are launched by the following command, terminal input, the

```
ros2 node list
```



```

yahboom@yahboom-virtual-machine:~/ros2_ws$ ros2 node list
/publisher_node
/subscriber_node
/turtlesim

```

It did start three nodes, so it was started successfully.

### 5.3. Source Code Analysis

```

#导入必要的库文件 # Import the necessary library files
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
node1 = IncludeLaunchDescription(
    PythonLaunchDescriptionSource([os.path.join(
        get_package_share_directory('topic_pkg'), 'launch'),
        '/multi_node_launch.py'])
)

```

- `os.path.join(get_package_share_directory('topic_pkg'))`: get the location of the feature package, where 'topic\_pkg' is the name of the feature package;
- `launch')`: indicates the folder where the launch file is stored under the feature package;
- `/multi_node_launch.py'`: the name of the launch file under the package folder, that is, `/multi_node_launch.py` in the example.

## 6. Configuring rosparam for launch file parameters

Terminal input.

```

cd ~/ros2_ws/src/topic_pkg/launch
gedit param_launch.py

```

Copy the following into that file.

```

from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument
from launch.substitutions import LaunchConfiguration, TextSubstitution
from launch_ros.actions import Node

def generate_launch_description():
    background_r_launch_arg = DeclareLaunchArgument(
        'background_r', default_value=TextSubstitution(text='0'))
    background_g_launch_arg = DeclareLaunchArgument(
        'background_g', default_value=TextSubstitution(text='225'))
    background_b_launch_arg = DeclareLaunchArgument(
        'background_b', default_value=TextSubstitution(text='0'))
    return LaunchDescription([
        background_r_launch_arg,
        background_g_launch_arg,
        background_b_launch_arg,
        Node(
            package='turtlesim',
            executable='turtlesim_node',
            name='sim',

```

```

        parameters=[{
'background_r':LaunchConfiguration('background_r'),
'background_g':LaunchConfiguration('background_g'),
'background_b':LaunchConfiguration('background_b'),
        }]
    )
])

```

## 6.1, Compile Workspace

Terminal input,

```

cd ~/ros2_ws
colcon build

```

After the compilation is complete, refresh the environment variables in the workspace.

```

source ~/ros2_ws/install/setup.bash

```

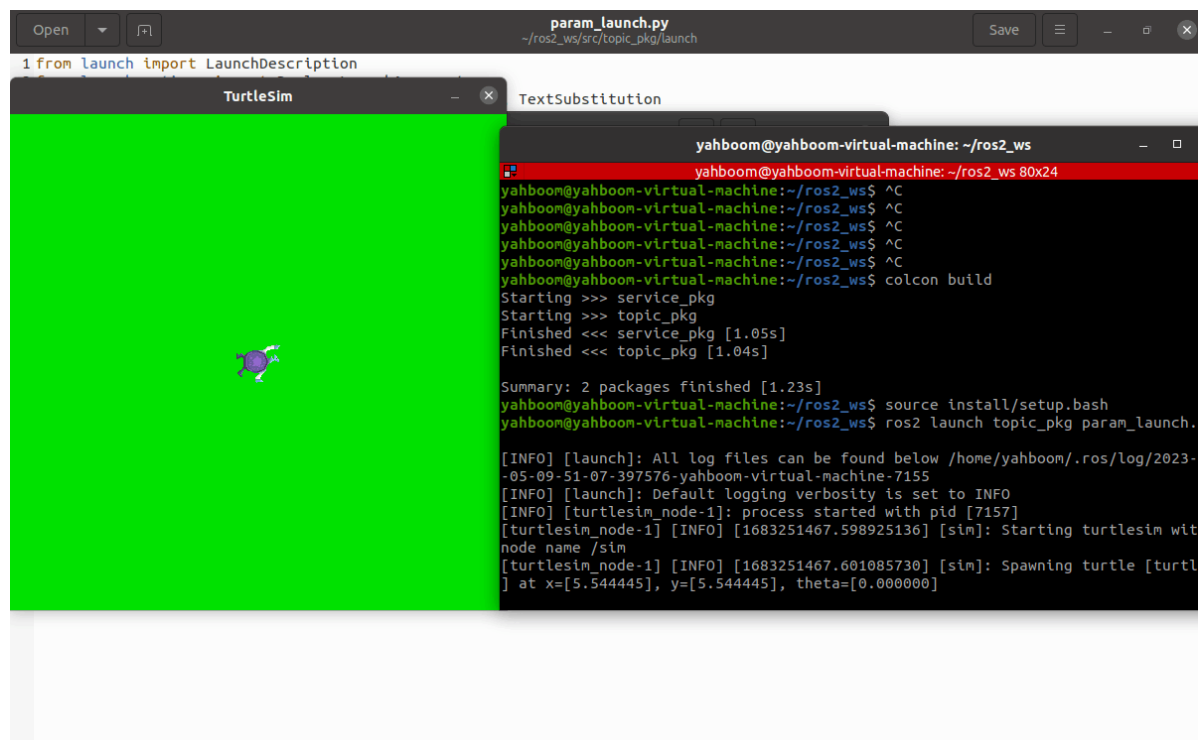
## 6.2. Run the program

Terminal input,

```

ros2 launch topic_pkg param_launch.py

```



After the program runs, it will load the set parameters and modify the default RGB parameters to change the color of the background plate.

## 6.3. Source Code Analysis

```
from launch.actions import DeclareLaunchArgument    # 声明launch文件内使用的
Argument类
# Declare the Argument class used within the launch file
background_r_launch_arg = DeclareLaunchArgument(
    'background_r', default_value=TextSubstitution(text='0')) # 创建一个Launch文
件内参数background_r
    # Create a Launch file within the parameter background_r
    'background_r', default_value=TextSubstitution(text='0')) # 创建一个Launch文
件内参数background_g
    # Create a Launch file within the parameter background_g
    'background_r', default_value=TextSubstitution(text='0')) # 创建一个Launch文
件内参数background_b
    # Create a Launch file within the parameter background_b
    background_r_launch_arg, # 调用以上创建的参数
    # Call the parameters created above
    background_g_launch_arg,
    background_b_launch_arg,
    parameters=[{                                # ROS参数列表
# ROS parameter list
    'background_r': LaunchConfiguration('background_r'), # 创建参数
background_r # Create the parameter background_r
    'background_g': LaunchConfiguration('background_g'), # 创建参数
background_g # Create the parameter background_g
    'background_b': LaunchConfiguration('background_b'), # 创建参数
background_b # Create the parameter background_b
```

argument and param both mean parameters, but argument is passed inside the launch file, while param is a parameter passed inside the node program.

## 7. launch file load parameter configuration table

First of all, create a new parameter table, terminal input.

```
cd ~/ros2_ws/src/topic_pkg
mkdir config
cd config
gedit turtle_config.yaml
```

Copy the following into the turtle\_config.yaml file.

```
sim:
  ros__parameters:
    background_r: 0
    background_g: 0
    background_b: 7
```

Save and exit, then modify the setup.py file to load the path to the parameter file, terminal input,

```
cd ~/ros2_ws/src/topic_pkg
gedit setup.py
```

```

1 from setuptools import setup
2 import os
3 from glob import glob
4 package_name = 'topic_pkg'
5
6 setup(
7     name=package_name,
8     version='0.0.0',
9     packages=[package_name],
10    data_files=[
11        ('share/ament_index/resource_index/packages',
12         ['resource/' + package_name]),
13        ('share/' + package_name, ['package.xml']),
14        (os.path.join('share', package_name, 'launch'), glob(os.path.join('launch', '*launch.py'))),
15        (os.path.join('share', package_name, 'config'), glob(os.path.join('config', '*.yaml'))),
16    ],
17    install_requires=['setuptools'],
18    zip_safe=True,
19    maintainer='yahboom',
20    maintainer_email='yahboom@todo.todo',
21    description='TODO: Package description',
22    license='TODO: License declaration',
23    tests_require=['pytest'],
24    entry_points={
25        'console_scripts': [
26            'publisher_demo = topic_pkg.publisher_demo:main',
27            'subscriber_demo = topic_pkg.subscriber_demo:main'
28        ],
29    },
30 )

```

Python Tab Width: 8 Ln 23, Col 30 INS

In the location shown in the image above, add the following.

```

(os.path.join('share', package_name, 'config'), glob(os.path.join('config',
'*.yaml'))),

```

Save and exit, and finally write the launch file and type in the terminal,

```

cd ~/ros2_ws/src/topic_pkg/launch
gedit param_config_launch.py

```

Copy the following into that file.

```

import os
from launch import LaunchDescription
from launch_ros.actions import Node
from ament_index_python.packages import get_package_share_directory
def generate_launch_description():
    config = os.path.join(
        get_package_share_directory('topic_pkg'),
        'config',
        'turtle_config.yaml'
    )
    return LaunchDescription([
        Node(
            package='turtlesim',
            executable='turtlesim_node',
            name='sim',
            parameters=[config]
        )
    ])

```

## 7.1, Compile Workspace

Terminal input,

```
cd ~/ros2_ws  
colcon build
```

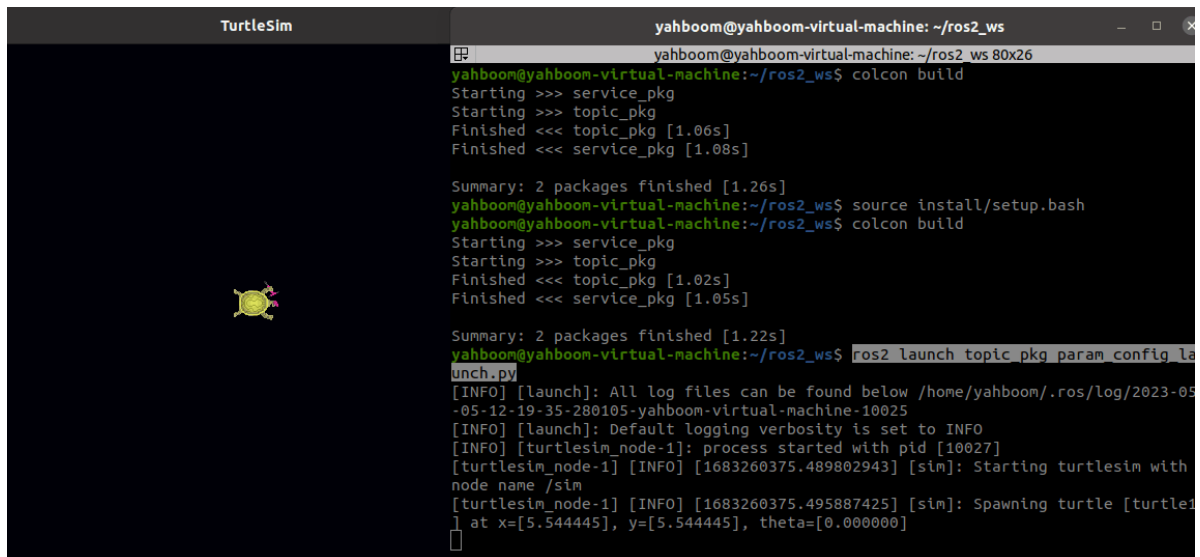
After the compilation is complete, refresh the environment variables in the workspace.

```
source ~/ros2_ws/install/setup.bash
```

## 7.2. Run the program

Terminal input,

```
ros2 launch topic_pkg param_config_launch.py
```



```
yahboom@yahboom-virtual-machine: ~/ros2_ws  
yahboom@yahboom-virtual-machine:~/ros2_ws$ colcon build  
Starting >>> service_pkg  
Starting >>> topic_pkg  
Finished <<< topic_pkg [1.06s]  
Finished <<< service_pkg [1.08s]  
  
Summary: 2 packages finished [1.26s]  
yahboom@yahboom-virtual-machine:~/ros2_ws$ source install/setup.bash  
yahboom@yahboom-virtual-machine:~/ros2_ws$ colcon build  
Starting >>> service_pkg  
Starting >>> topic_pkg  
Finished <<< topic_pkg [1.02s]  
Finished <<< service_pkg [1.05s]  
  
Summary: 2 packages finished [1.22s]  
yahboom@yahboom-virtual-machine:~/ros2_ws$ ros2 launch topic_pkg param_config_la  
unch.py  
[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2023-05-12-19-35-280105-yahboom-virtual-machine-10025  
[INFO] [launch]: Default logging verbosity is set to INFO  
[INFO] [turtlesim_node-1]: process started with pid [10027]  
[turtlesim_node-1] [INFO] [1683260375.489802943] [sim]: Starting turtlesim with node name /sim  
[turtlesim_node-1] [INFO] [1683260375.495887425] [sim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]  
□
```

Running the program can get a small turtle and the background plate color is set to black by the parameter.

## 7.3. Source Code Analysis

```
#找到参数文件位置 # Find the location of the parameter file  
config = os.path.join(  
    get_package_share_directory('topic_pkg'),  
    'config',  
    'turtle_config.yaml'  
)  
  
#加载参数文件 #Load the parameter file  
parameters=[config]
```

Let's look at the parameter file turtle\_config.yaml.

```
sim:  
  ros__parameters:  
    background_r: 0  
    background_g: 0  
    background_b: 7
```

The parameter file is located at: `~/ros2_ws/src/topic_pkg/config`

- `sim`: the name of the node
- `ros__parameters`: the ros parameters, which are fixed here.
- `background_r` parameter name, followed by the value of the parameter set value