

## 5. ROS2launch file is launched

---

In ROS2, launch is used to start multiple nodes and configure program running parameters. The launch file format of ROS2 is xml, yaml, and python. This lesson uses the launch file in python format as an example, which is more flexible than the other two formats:

- python has many libraries that can be used in startup files;
- ROS2 generic launch features and specific launch features are written in Python and thus have access to launch features that XML and YAML may not expose;

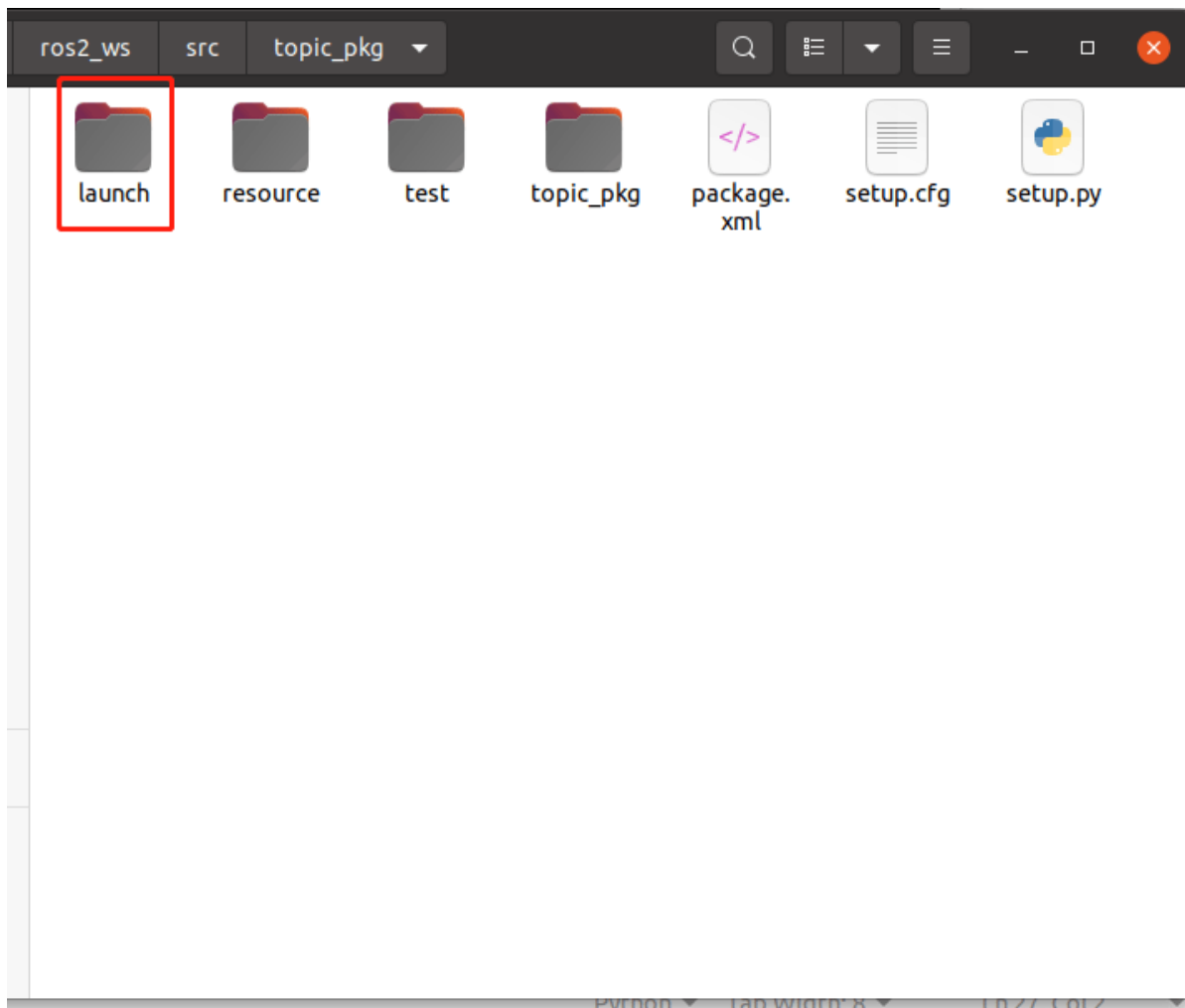
Using python language to write ROS2 launch file, the most important thing is to abstract each node, file, script, etc. into an action, with a unified interface to start, the main structure is:

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

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

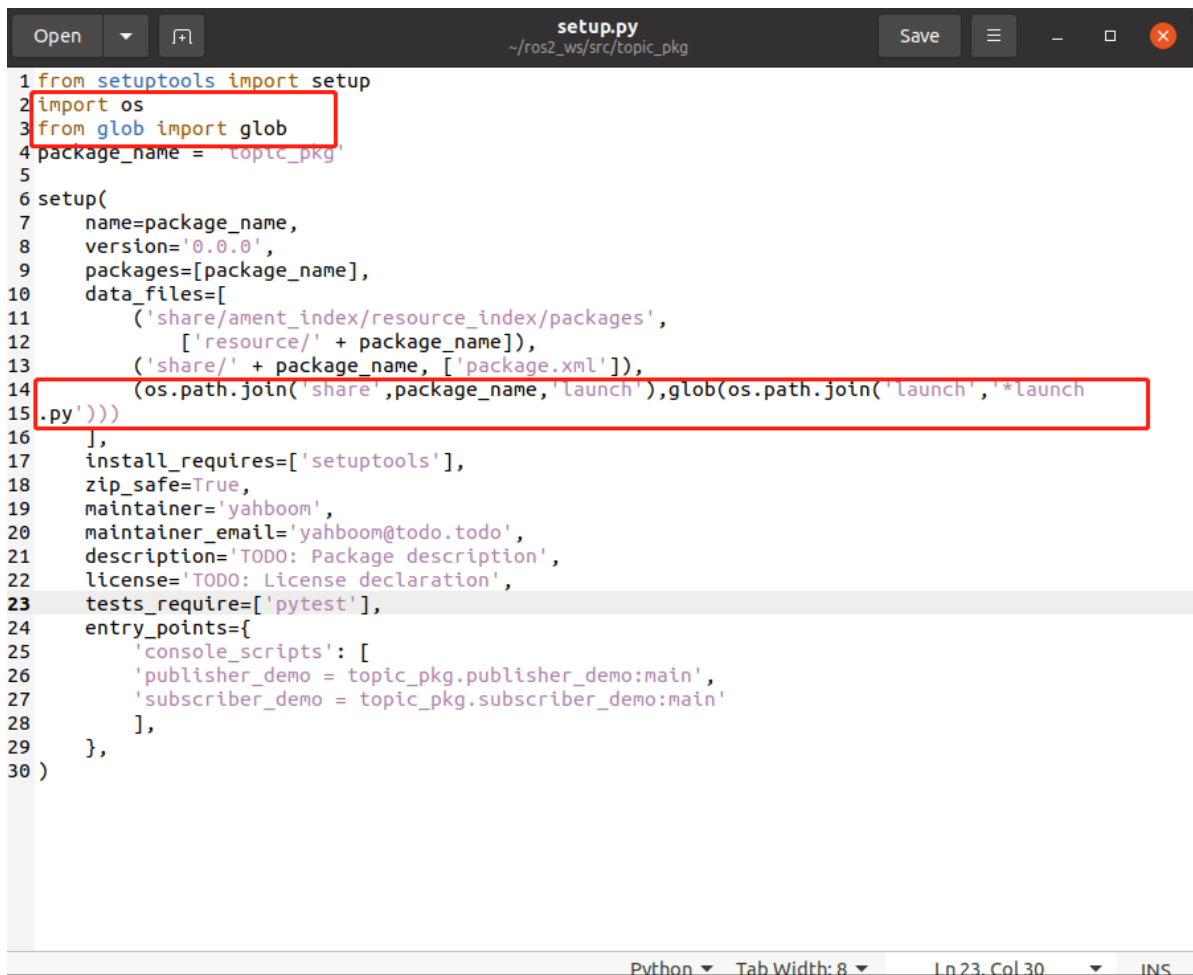
We create a new folder under the path of the function package created before to store the launch file, terminal input,

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



The launch file is usually named `LaunchName_launch.py`, where `LaunchName` is custom and `_launch.py` is usually considered fixed. You need to modify the `setup.py` file in the function `package` to add the file in the launch path, so that the `.py` file can be generated by compiling.

```
#1、导入相关的头文件
#1, import the relevant header file
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 data_files list, add the launch path 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 launch

Terminal input,

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

Copy the following into the 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 the workspace

Terminal input,

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

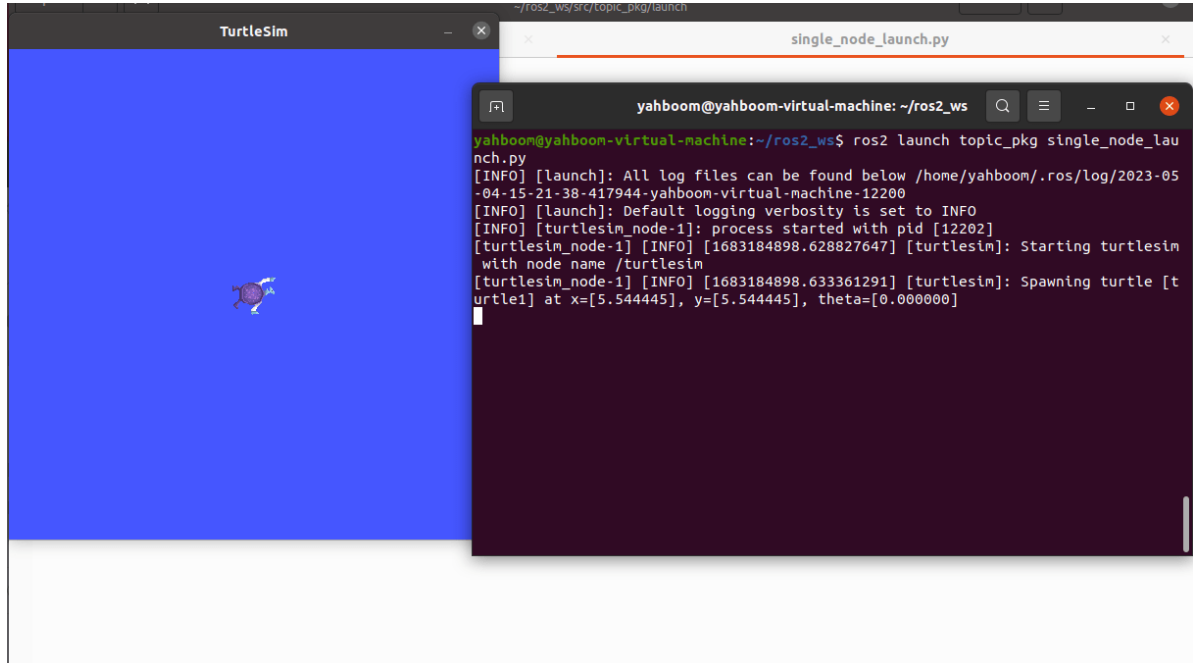
After compiling, refresh the environment variables in the workspace,

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

## 2.2. Run the program

Terminal input,

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



After the program runs, it will run on the node of the turtle.

## 2.3 Source code analysis

### 1). Import relevant libraries

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

After the program runs, it will run on the node of the turtle.

### 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
```

Defines a variable `turtle_node` as the return value of a `Node` start, calls the node function, launches two important parameters, `package` and `executable`.

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

The variable `launch_description` is then defined as the return value of the `LaunchDescription` function, which can be added later if multiple nodes are started.

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

Finally, return launch\_description.

### 3. Write multiple Node launches

Terminal input,

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

Copy the following into the 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 the workspace

Terminal input,

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

After compiling, 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
```

The terminal does not print content, we can check which nodes start to verify whether there is a successful start, terminal input,

```
ros2 node list
```

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

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

### 3.3 Source code analysis

It's roughly the same as `simple_node_launch.py`, but with an extra node and an extra node in `launch_description = LaunchDescription([pub_node,sub_node])`.

## 4. Topic name mapping in launch file

Terminal input,

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

Copy the following into the 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 the workspace

Terminal input,

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

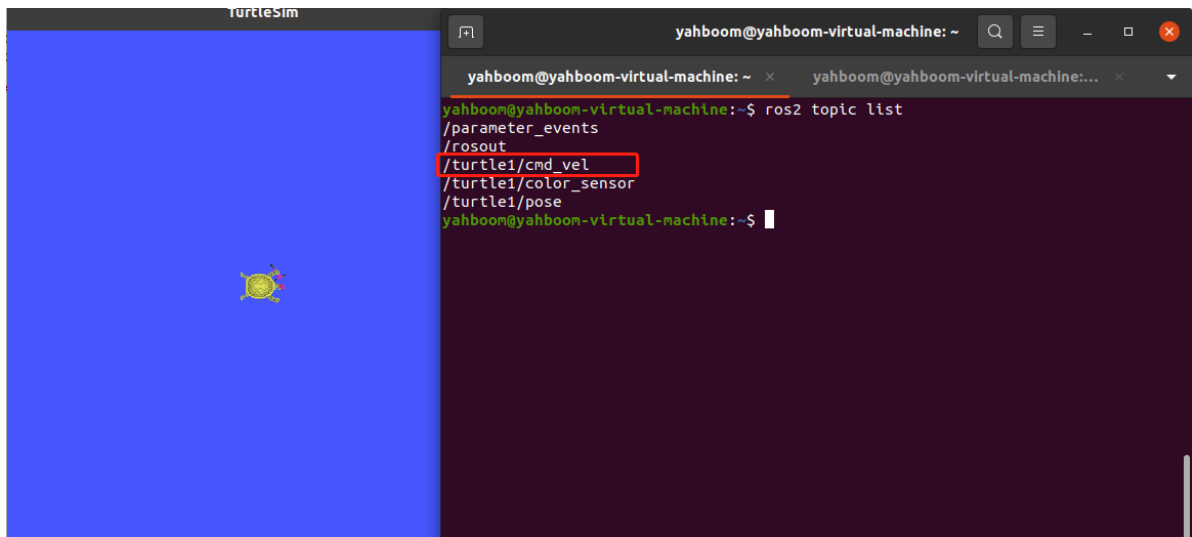
After compiling, refresh the environment variables in the workspace,

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

### 4.2. Run the program

Let's see what the speed of the baby turtle is before we remap the topic what the topic name is, terminal input,

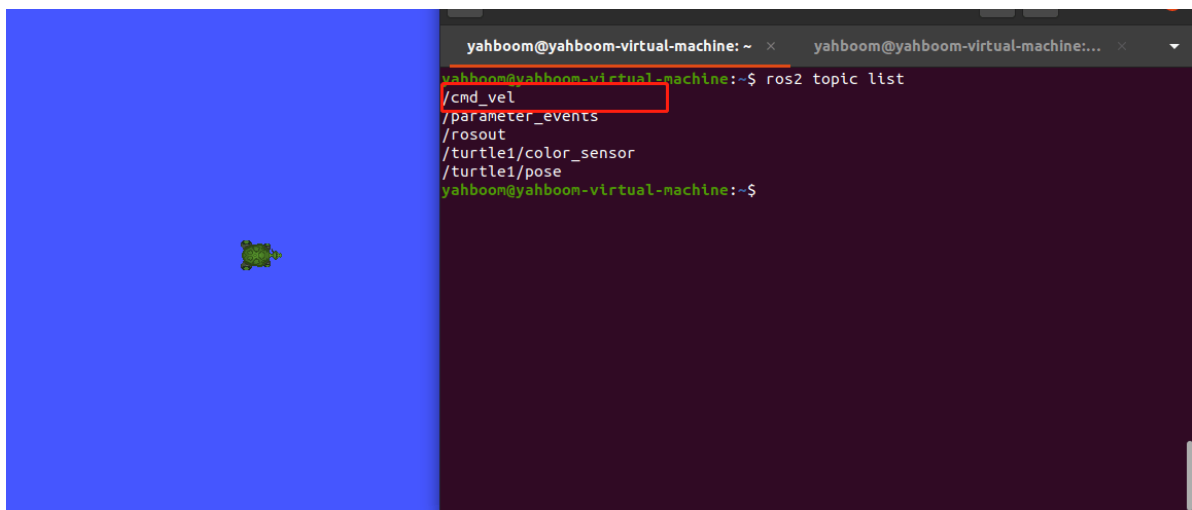
```
ros2 launch topic_pkg single_node_launch.py
ros2 topic list
```



The topic here is /turtle1/cmd\_vel.

Run the program after remapping the topic, see what the speed topic name of the turtle subscription is, terminal input,

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



As can be seen from the figure above, the speed topic name is remapped, and the mapped speed topic name of the baby turtle is /cmd\_vel.

### 4.3 Source code analysis

The original single\_node\_launch.py has been modified, mainly adding the following parts:

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

This is where the original /turtle1/cmd\_vel is remapped to /cmd\_vel. The original topic name in the front, and the topic name we want to change into the back.

## 5. launch file launch the launch file

Terminal input,

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

Copy the following into the 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 the workspace

Terminal input,

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

After compiling, refresh the environment variables in the workspace,

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

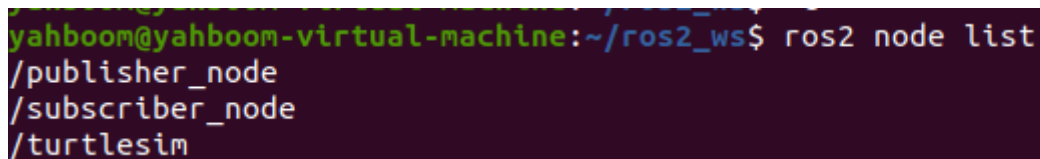
## 5.2. Run the program

Terminal input,

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

The launch file will contain two launch files, simple\_node\_launch.py and multi\_node\_launch.py. You can check whether the nodes of these launch files are started by the following command, terminal input,

```
ros2 node list
```



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

Three nodes were indeed started, so it was successful.



## 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'))` : obtains the location of the function package, where 'topic\_pkg' is the name of the function package.
- `launch')` : indicates the folder where the launch file is stored under the function package.
- `/multi_node_launch.py'` : indicates the name of the launch file under the launch file in the function package folder, which is `/multi_node_launch.py` in the example.

## 6. Configure rosparam with launch file parameters

Terminal input,

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

Copy the following into the 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 the workspace

Terminal input,

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

After compiling, 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
```

```
param_launch.py
~/ros2_ws/src/topic_pkg/launch

1 from launch import LaunchDescription
...

yahboom@yahboom-virtual-machine: ~/ros2_ws
yahboom@yahboom-virtual-machine: ~/ros2_ws 80x24
yahboom@yahboom-virtual-machine:~/ros2_ws$ ^C
yahboom@yahboom-virtual-machine:~/ros2_ws$ ^C
yahboom@yahboom-virtual-machine:~/ros2_ws$ ^C
yahboom@yahboom-virtual-machine:~/ros2_ws$ ^C
yahboom@yahboom-virtual-machine:~/ros2_ws$ ^C
yahboom@yahboom-virtual-machine:~/ros2_ws$ colcon build
Starting >>> service_pkg
Starting >>> topic_pkg
Finished <<< service_pkg [1.05s]
Finished <<< topic_pkg [1.04s]

Summary: 2 packages finished [1.23s]
yahboom@yahboom-virtual-machine:~/ros2_ws$ source install/setup.bash
yahboom@yahboom-virtual-machine:~/ros2_ws$ ros2 launch topic_pkg param_launch.

[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2023-
-05-09-51-07-397576-yahboom-virtual-machine-7155
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [turtlesim_node-1]: process started with pid [7157]
[turtlesim_node-1] [INFO] [1683251467.598925136] [sim]: Starting turtlesim wit
node name /sim
[turtlesim_node-1] [INFO] [1683251467.601085730] [sim]: Spawning turtle [turtl
] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

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

## 6.3 Source code analysis

```
from launch.actions import DeclareLaunchArgument    # 声明launch文件内使用的
Argument类
# Declares the Argument class used in 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
```

```

# Creates a Launch file with the background_r parameter
'background_r', default_value=TextSubstitution(text='0')) # 创建一个Launch文件内参数background_r
# Creates a Launch file with the background_g parameter
'background_r', default_value=TextSubstitution(text='0')) # 创建一个Launch文件内参数background_b
# Creates a Launch file with the background_b argument
background_r_launch_arg, # 调用以上创建的参数
# calls 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 parameter background_r
'background_g': LaunchConfiguration('background_g'), # 创建参数
background_g
# Create parameter background_g
'background_b': LaunchConfiguration('background_b'), # 创建参数
background_b
# Create parameter background_b

```

argument and param are both parameters, but argument is passed in the launch file, and param is passed in the node program.

## 7. launch file load parameter configuration table

First create a parameter table, terminal input,

```

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

```

Copy the contents 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, the path to load 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 position shown in the above picture, add the following,

```

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

```

Save and exit, finally write the launch file, terminal input,

```

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

```

Copy the following into the 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 the workspace

Terminal input,

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

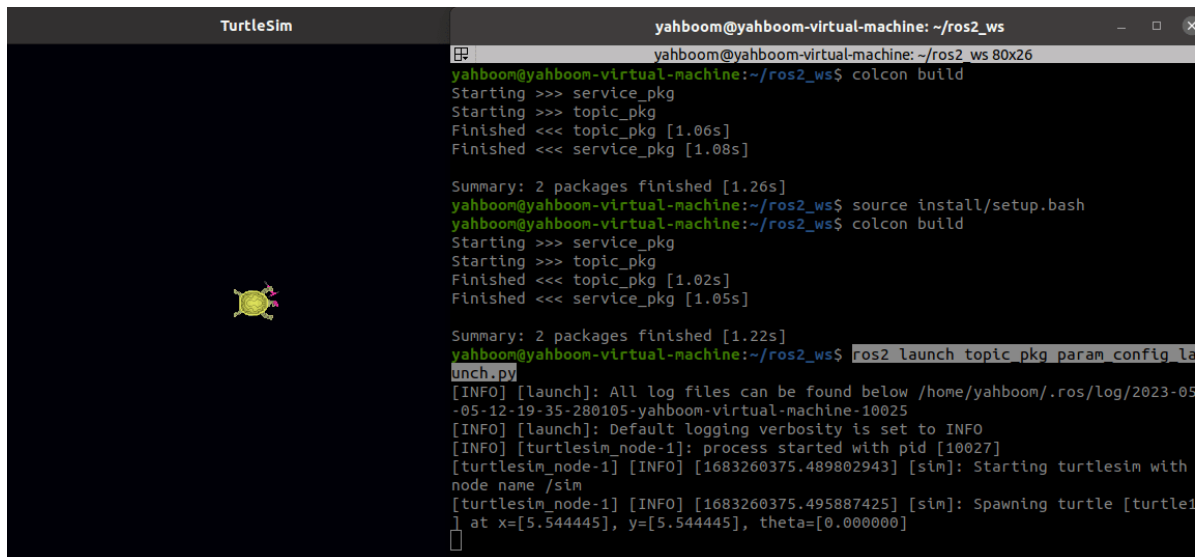
After compiling, refresh the environment variables in the workspace,

```
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]  
□
```

Run the program to get the turtle and the background color is set to black by parameter.

## 7.3 Source code analysis

```
#找到参数文件位置  
# Find the parameter file location  
config = os.path.join(  
    get_package_share_directory('topic_pkg'),  
    'config',  
    'turtle_config.yaml'  
)  
#加载参数文件  
# Load 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 location of the parameter file is `~/ros2_ws/src/topic_pkg/config`

- `sim`: indicates the node name
- `ros__parameters`: indicates the ros parameter, which is fixed
- `background_r` Specifies the parameter name. The following value is the parameter setting value