

## 5、ROS2launch file starts

---

In ROS2, launch is used for multi-node startup and configuring program running parameters, and ROS2's launch file formats include XML, Yaml and Python formats. This lesson takes the launch file in Python format as an example, compared to the other two formats, the Python format is more flexible:

- Python has a large number of libraries that can be used in startup files;
- ROS2 generic boot features and specific boot features are written in Python, so XML and YAML boot features that may not be exposed can be accessed;

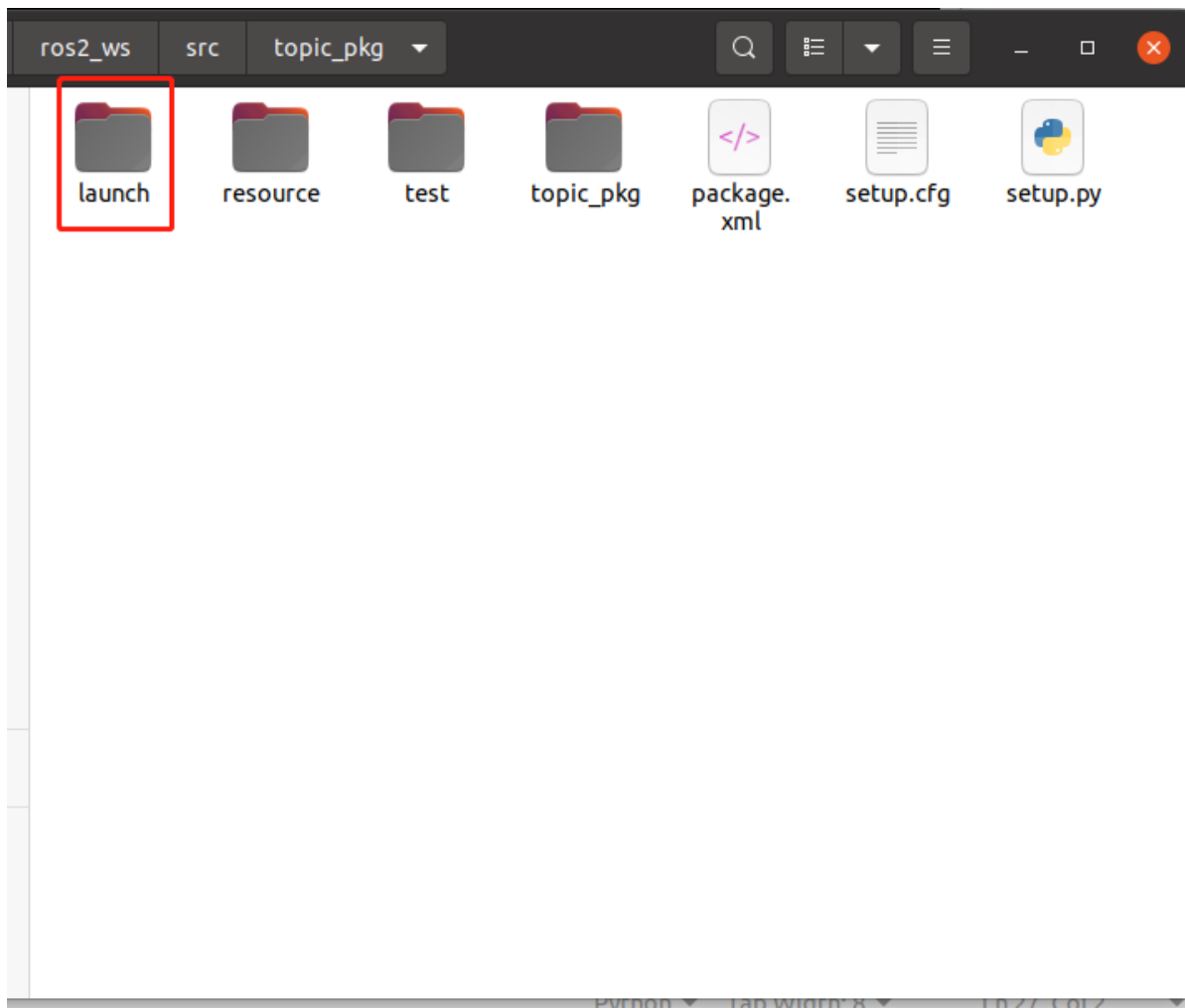
Using python language to write ROS2 launch files, the most important thing is to abstract each node, file, script, etc. into an action, with a unified interface to start, the main structure is,

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

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

Under the path of the function package created earlier, we create a new folder to store the launch file and terminal input.

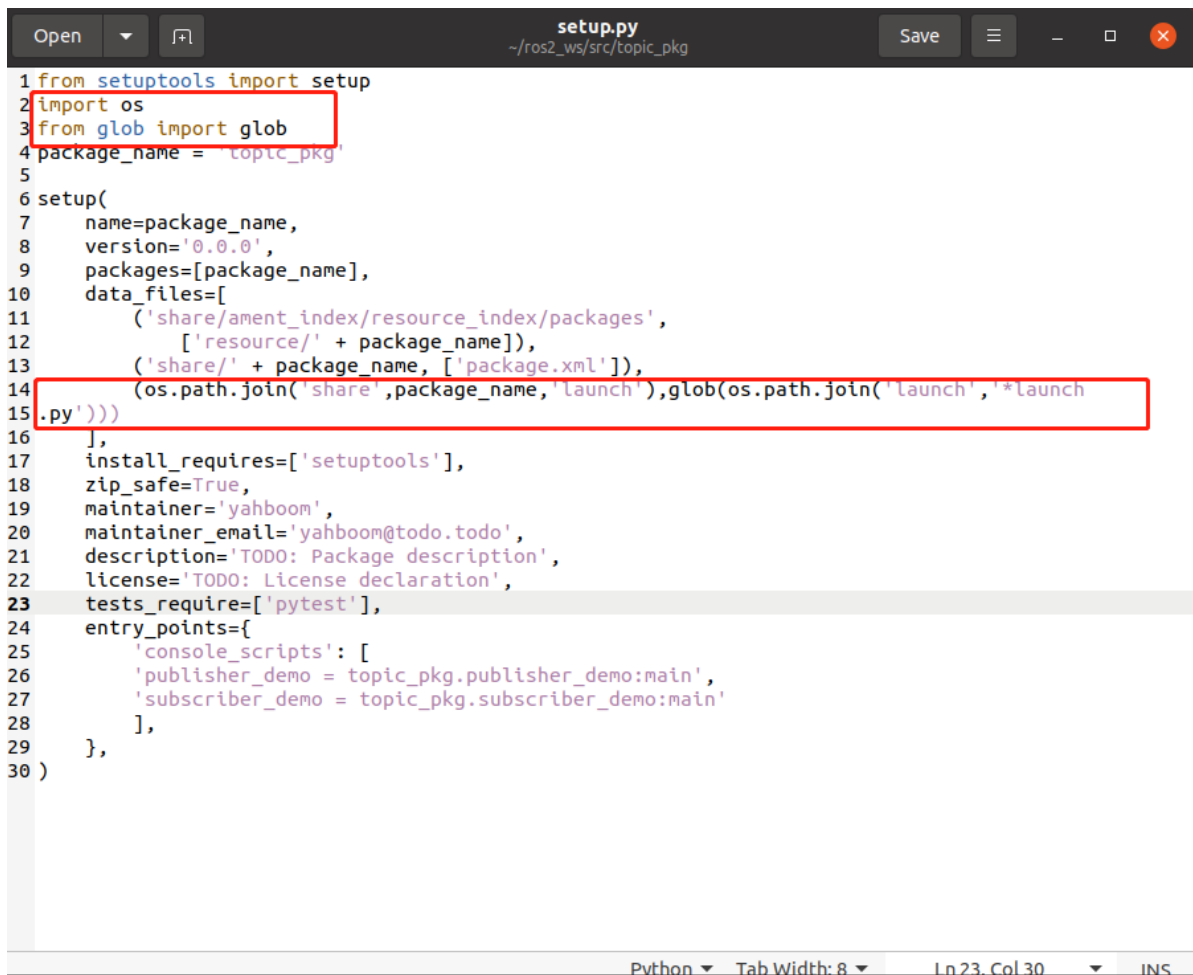
```
cd ~/ros2_ws/src/topic_pkg  
mkdir launch
```



Launch files are often named in `LaunchName_launch.py`, where `LaunchName` is custom, `_launch.py` is often considered fixed. You need to modify the `setup.py` file under the function `pack`, modify the content to add the file under the launch path, and compile to generate the executed `.py` file.

```
#1、Import the relevant header files
import os
from glob import glob

#2、In the list of data_files, add the launch path and the launch.py files 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 launch for a single node node

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
```

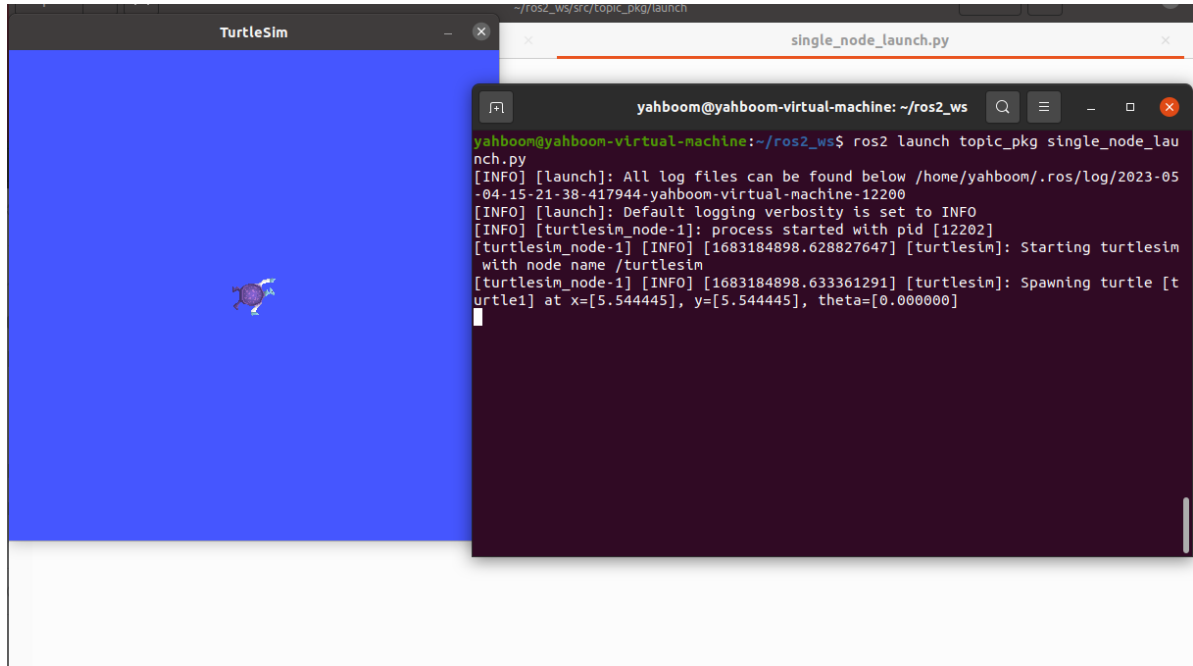
After the compilation is completed, refresh the environment variables of the workspace.

```
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 the baby turtle node.

## 2.3、 Source code analysis

### 1)、 Import related libraries

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

### 2)、 Defines a function generate\_launch\_description and returns a launch\_description

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

Define a variable turtle\_node return value for starting as a node, call the Node function, and launch two important parameters, package and executable.

- package: Indicates the function package, representing the name of the feature package.
- executable: indicates the executed program, the name of the executable program.

Then define a variable `launch_description` as the return value after calling the `LaunchDescription` function, and add it after multiple nodes are started.

```
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 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 the compilation is completed, refresh the environment variables of the workspace.

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

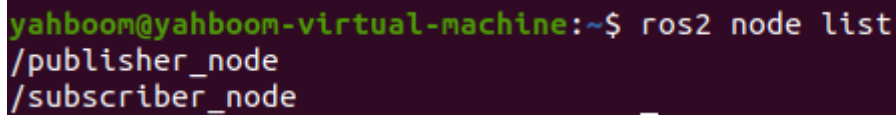
#### 3.2、 Run the program

terminal input,

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

The terminal does not print content, we can see which nodes are started to verify that there is a successful startup, the terminal inputs,

```
ros2 node list
```

A terminal window with a dark purple background. The prompt is 'yahboom@yahboom-virtual-machine:~\$'. The command 'ros2 node list' has been entered, and the output shows two nodes: '/publisher\_node' and '/subscriber\_node' on separate lines.

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

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

### 3.3、 Source code parsing

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

## 4、 Topic name mapping in launch files

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
```

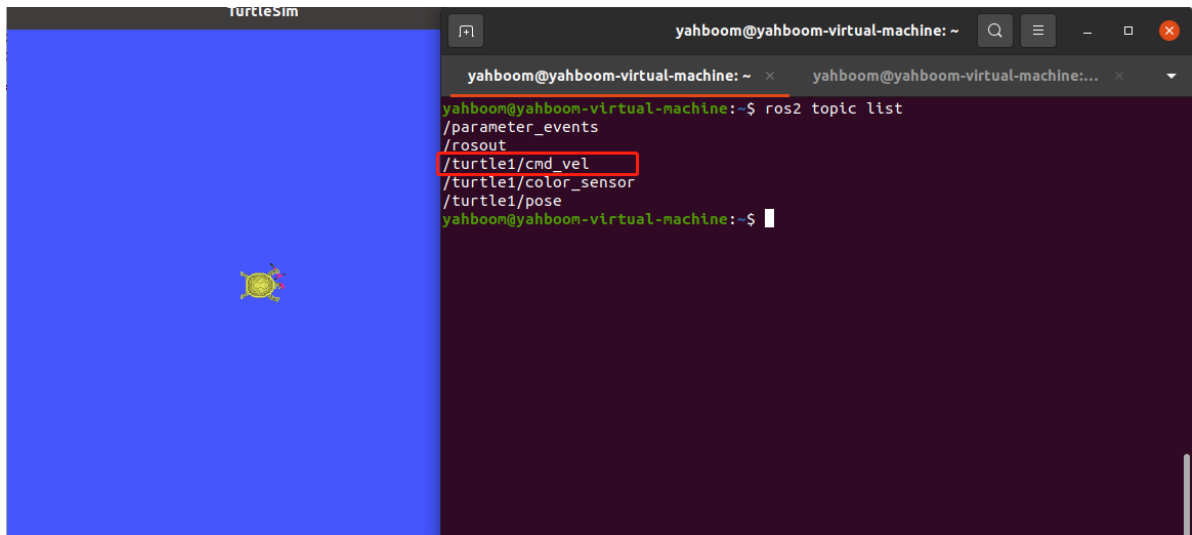
After the compilation is completed, refresh the environment variables of the workspace.

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

### 4.2、 Run the program

Let's first see what is the name of the little turtle's speed topic before remapping the topic,  
terminal input,

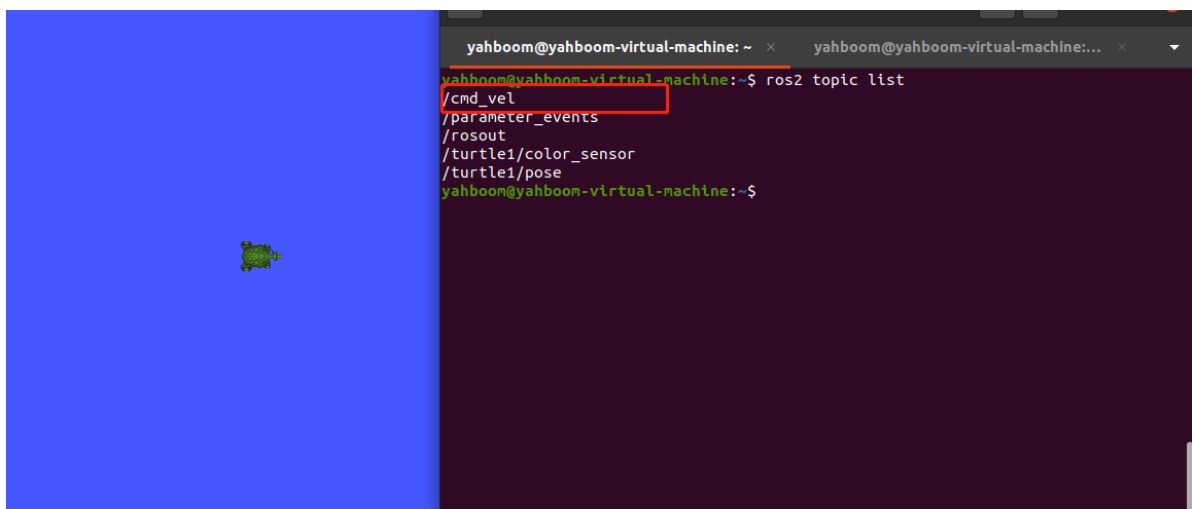
```
ros2 launch topic_pkg single_node_launch.py
ros2 topic list
```



The topic here is /turtle1/cmd\_vel.

Then run the program after remapping the topic, see what is the name of the speed topic that the little turtle subscribes to, and enter it in the terminal,

```
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 baby turtle speed topic name is /cmd\_vel.

### 4.3、 Source code analysis

Modifications have been made on the basis of the original single\_node\_launch.py, mainly adding the following parts,

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

This is to remap the original /turtle1/cmd\_vel to /cmd\_vel. The original topic name in front is the topic name we want to change to.

## 5、 Edit 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 the compilation is completed, refresh the environment variables of 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 two launch files, simple\_node\_launch.py and multi\_node\_launch.py. You can check whether the node of these launch files is started by the following command, terminal input,

```
ros2 node list
```



```

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

```

It is true that three nodes are started, so the startup is successful.

### 5.3、 Source code analysis

```

#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'))`: Gets the location of the feature pack, where 'topic\_pkg' is the name of the feature pack;
- `launch')`: indicates the folder that stores the launch file under the function pack;
- `/multi_node_launch.py'`: indicates the name of the launch file under the launch file in the package folder, which is `/multi_node_launch.py` in the example.

## 6、 Launch file parameter configuration rosparam

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 the compilation is completed, refresh the environment variables of the workspace.

```

source ~/ros2_ws/install/setup.bash

```

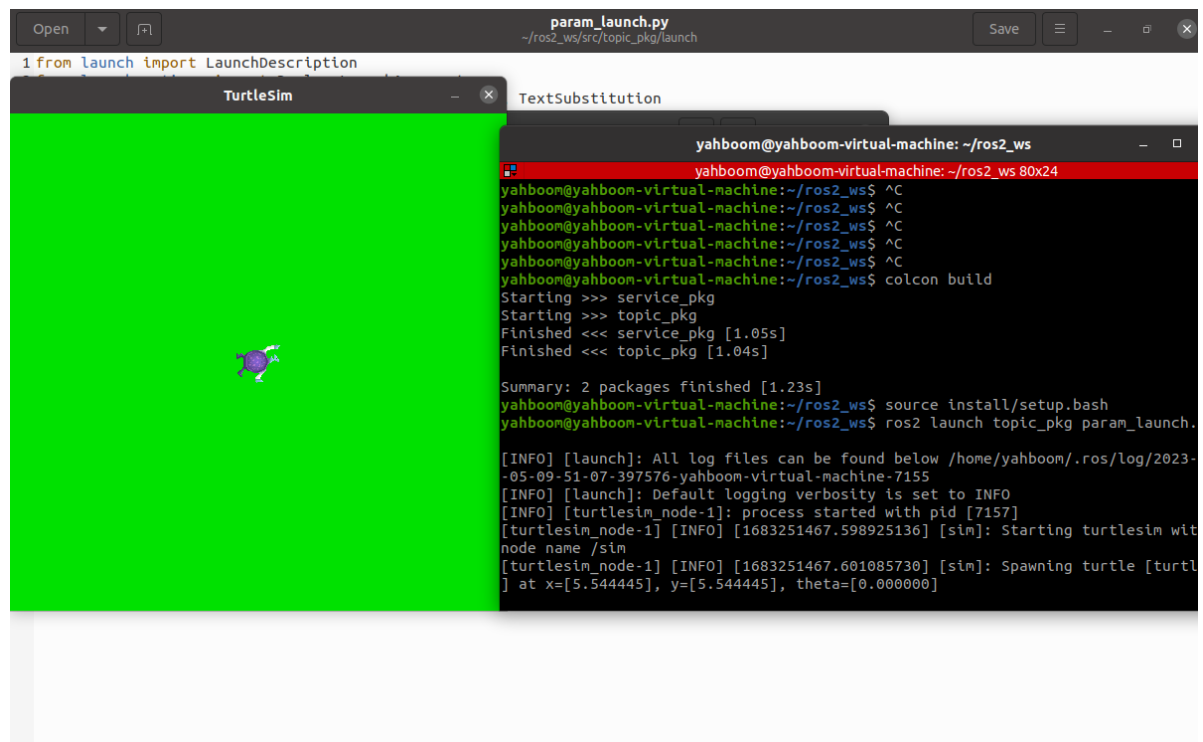
## 6.2、 Run the program

terminal input,

```

ros2 launch topic_pkg param_launch.py

```



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

## 6.3、Source code parsing

```
from launch.actions import DeclareLaunchArgument    #Declare the Argument class
used in the launch file
background_r_launch_arg = DeclareLaunchArgument(
    'background_r', default_value=TextSubstitution(text='0')) #Create a Launch
file with parameter background_r
    'background_r', default_value=TextSubstitution(text='0')) #Create a Launch
file with parameter background_g
    'background_r', default_value=TextSubstitution(text='0')) #Create a Launch
file with parameter background_b
    background_r_launch_arg, #Call the parameter created above
    background_g_launch_arg,
    background_b_launch_arg,
    parameters=[{                                #ROS parameter
list
        'background_r': LaunchConfiguration('background_r'),    #Create
parameters background_r
        'background_g': LaunchConfiguration('background_g'),    #Create
parameters background_g
        'background_b': LaunchConfiguration('background_b'),    #Create a
parameter background_b
```

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

## 7、 launch file loads the parameter configuration table

First create a new parameter table, terminal input,

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

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

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

Exit after saving, then modify the setup.py file, the path to the loading 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 image above, add the following,

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

Exit after saving, 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
```

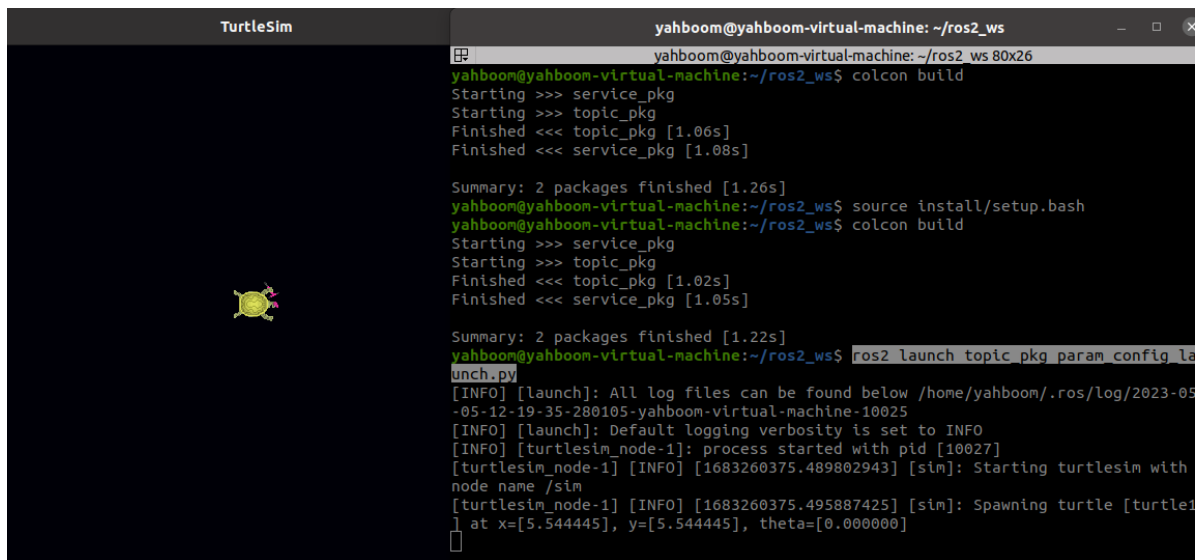
After the compilation is completed, refresh the environment variables of the workspace.

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

## 7.2、 Run the program

terminal input,

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



The screenshot shows a terminal window with two panes. The left pane, titled 'TurtleSim', displays a small green turtle icon on a black background. The right pane shows the terminal output of the command 'ros2 launch topic\_pkg param\_config\_launch.py'. The output includes the following text:

```
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 baby turtles and the background plate color is set to black by parameters.

## 7.3、 Source code parsing

```
#Locate the parameter file location  
config = os.path.join(  
    get_package_share_directory('topic_pkg'),  
    'config',  
    'turtle_config.yaml'  
)  
  
#Load the parameters 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 location is: `~/ros2_ws/src/topic_pkg/config`

- `sim:` indicates the node name
- `ros__parameters:` Indicates the ROS parameter, which is fixed here
- `background_r` the parameter name, followed by the value set by the parameter