14.ROS2 parameter service case

1. Parameter introduction

Similar to global variables in C++ programming, it is easy to share certain data in multiple programs. The parameters are global dictionaries in the ROS robot system, which can share data in multiple nodes.

In the ROS system, parameters exist in the form of a global dictionary. What is a dictionary? Just like a real dictionary, it is composed of names and values, also called keys and values, to synthesize key values. Or we can also understand that, just like parameters in programming, there is a parameter name, followed by an equal sign, followed by the parameter value. When using it, just access this parameter name.

The characteristics of parameters are very rich. For example, if a node shares a parameter, other nodes can access it. If a node modifies the parameter, other nodes can know it immediately and obtain the latest value.

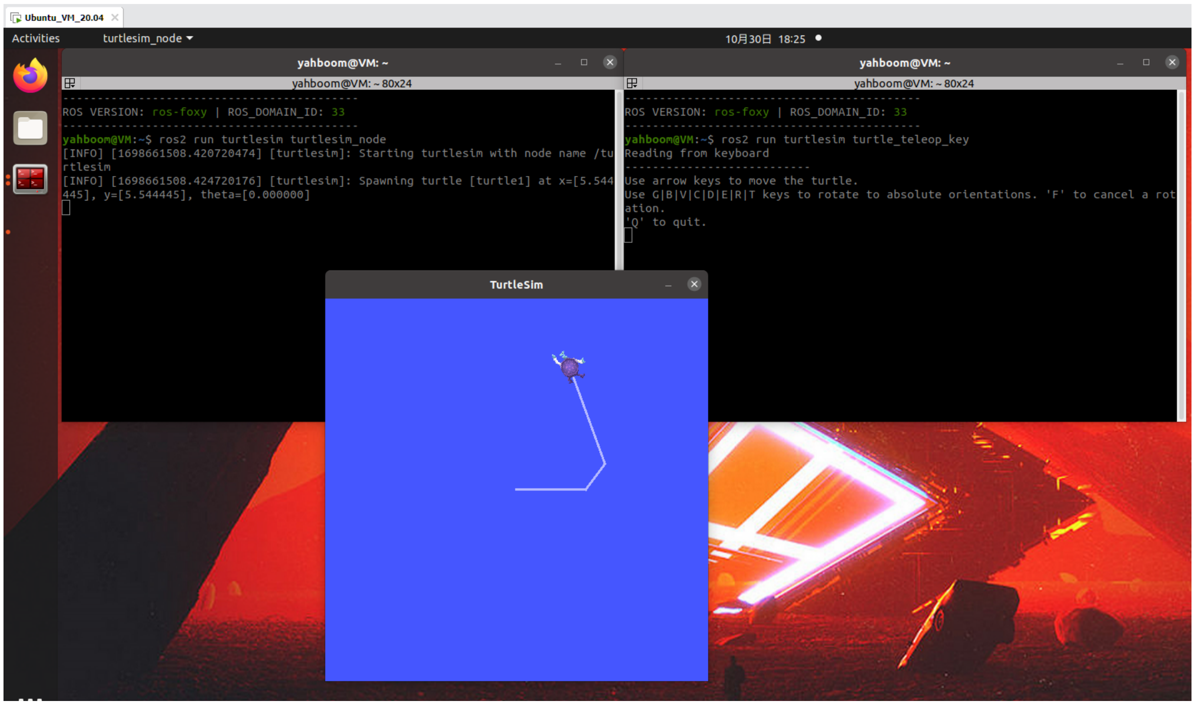
2. Parameters in the little turtle routine

In the routine of the little turtle, the emulator also provides a lot of parameters. Let's go through this routine to get familiar with the meaning of the parameters and how to use the command line.

Since the GUI is to be displayed here, for the convenience of operation, the following case is demonstrated in the virtual machine accompanying the tutorial.

1. Start two terminals in the virtual machine and run the turtle emulator and keyboard control node respectively:

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

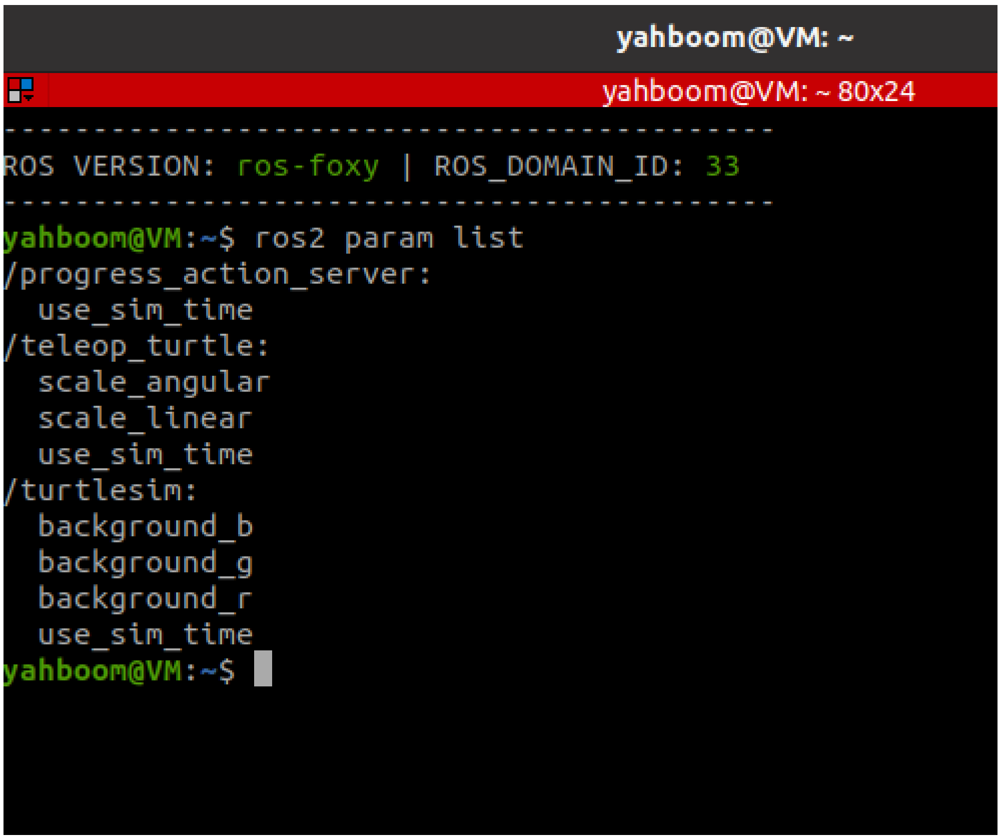


```
------------------------------------------
ROS VERSION: ros-foxy | ROS_DOMAIN_ID: 33
------------------------------------------
yahboom@VM:~$ ros2 run turtlesim turtlesim_node
[INFO] [1698661508.420720474] [turtlesim]: Starting turtlesim with node name /tu
rtlesim
[INFO] [1698661508.424720176] [turtlesim]: Spawning turtle [turtle1] at x=[5.544
445], y=[5.544445], theta=[0.000000]
```

```
------------------------------------------
ROS VERSION: ros-foxy | ROS_DOMAIN_ID: 33
------------------------------------------
yahboom@VM:~$ ros2 run turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rot
ation.
'Q' to quit.
```

2. Start a terminal on the virtual machine and use the following command to view the parameter list

```
ros2 param list
```



```
------------------------------------------
ROS VERSION: ros-foxy | ROS_DOMAIN_ID: 33
------------------------------------------
yahboom@VM:~$ ros2 param list
/progress_action_server:
  use_sim_time
/teleop_turtle:
  scale_angular
  scale_linear
  use_sim_time
/turtlesim:
  background_b
  background_g
  background_r
  use_sim_time
yahboom@VM:~$ 
```

3. Parameter query and modification

If you want to query or modify the value of a parameter, you can follow the param command with the get or set subcommand:

```
ros2 param describe turtlesim background_b   # View the description information
of a parameter
ros2 param get turtlesim background_b        # Query the value of a certain
parameter
ros2 param set turtlesim background_b 10     # Modify the value of a certain
parameter
```

4. Saving and loading parameter files

It is too troublesome to query/modify parameters one by one. It is better to try parameter files. Parameter files in ROS use yaml format. You can follow the param command with the dump subcommand to save the parameters of a certain node to the file.Or load all the contents in a parameter file at once through the load command:

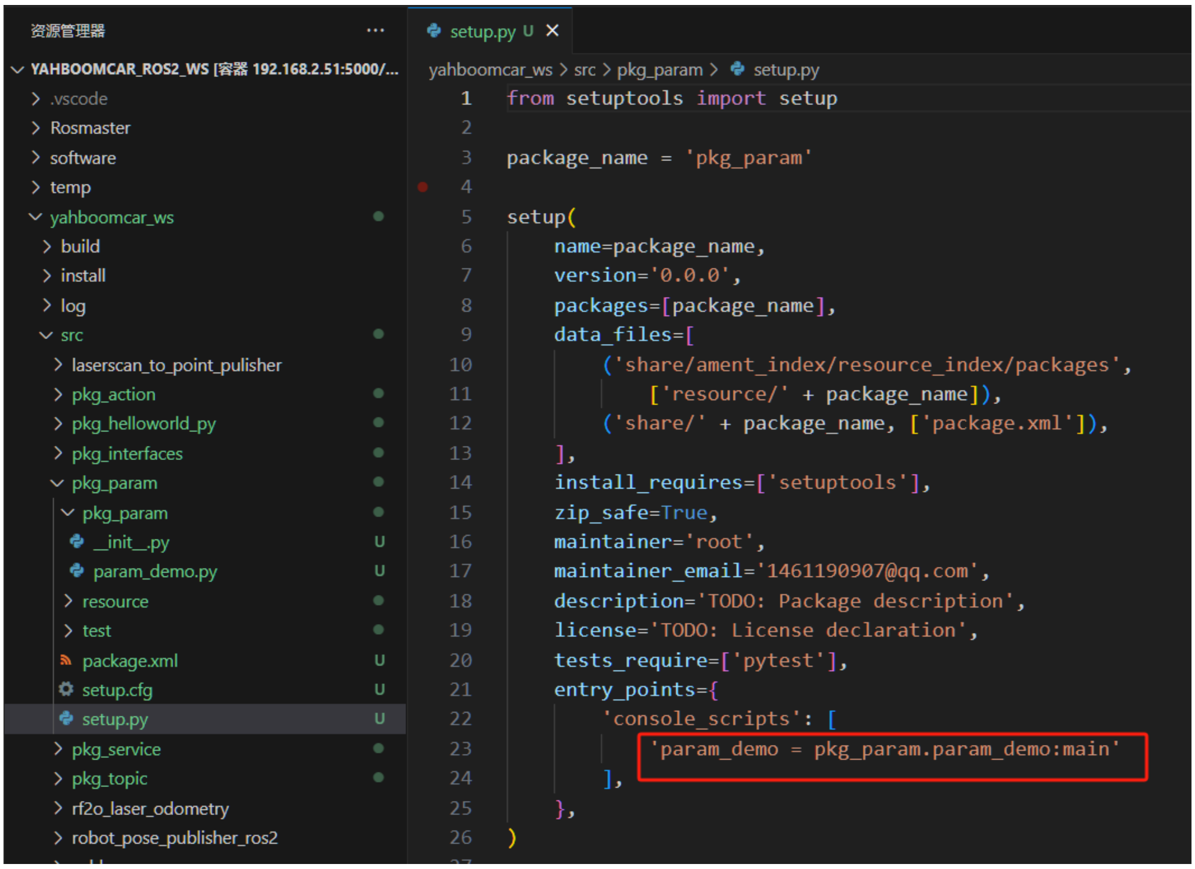
```
ros2 param dump turtlesim >> turtlesim.yaml  # Save the parameters of a node to a
parameter file
ros2 param load turtlesim turtlesim.yaml     # Load all parameters in a file at
once
```

3. Parameter case

3.1. Create new function package

```
cd ~/yahboomcar_ros2_ws/yahboomcar_ws/src
ros2 pkg create pkg_param --build-type ament_python --dependencies rclpy --node-
name param_demo
```

After executing the above command, the pkg_param function package will be created, and a param_demo node will be created, and the relevant configuration files have been configured.



```python
from setuptools import setup

package_name = 'pkg_param'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='root',
    maintainer_email='1461190907@qq.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'param_demo = pkg_param.param_demo:main'
        ],
    },
)
```

3.2. Code implementation

Next edit [param_demo.py] to implement the publisher’s functions and add the following code:

```python
import rclpy                                         # ROS2 Python interface library
from rclpy.node import Node                        # ROS2 Node class

class ParameterNode(Node):
    def __init__(self, name):
        super().__init__(name)                                           # ROS2 node
parent class initialization
        self.timer = self.create_timer(2, self.timer_callback)    # Create a
timer (period in seconds, callback function to be executed regularly)
        self.declare_parameter('robot_name', 'muto')              # Create a
parameter and set its default value

    def timer_callback(self):                                         # Create a
callback function that is executed periodically by the timer
        robot_name_param =
self.get_parameter('robot_name').get_parameter_value().string_value   # Read the
value of the parameter from the ROS2 system
        self.get_logger().info('Hello %s!' % robot_name_param)      # Output log
information and print the read parameter values

def main(args=None):                                    # ROS2 node main entrance
main function
    rclpy.init(args=args)                               # ROS2 Python interface
initialization
    node = ParameterNode("param_declare")               # Create ROS2 node objects
and initialize it
```

```
    rclpy.spin(node)                                  # Loop wait for ROS2 to
exit
    node.destroy_node()                               # Destroy node object
    rclpy.shutdown()                                  # Close the ROS2 Python
interface
```

3.3. Compile workspace

```
cd ~/yahboomcar_ros2_ws/yahboomcar_ws
colcon build --packages-select pkg_param
source install/setup.bash
```

3.4. Run the program

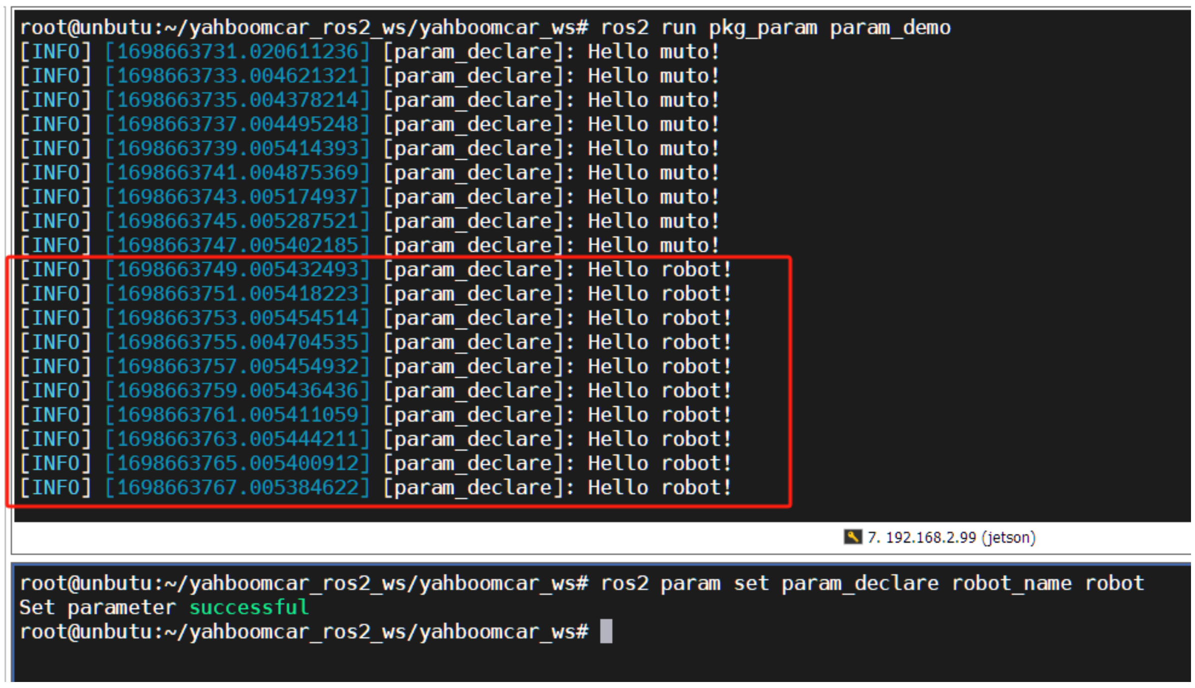
Open a terminal:

```
ros2 run pkg_param param_demo
```

Open another terminal:

```
# Set robot_name to robot
ros2 param set param_declare robot_name robot
```

You can see the log information printed in a loop in the terminal. "muto" is a parameter value that we set by default. The parameter name is "robot_name". After modifying this parameter through the command line, you can see that the terminal has also changed.



```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 run pkg_param param_demo
[INFO] [1698663731.020611236] [param_declare]: Hello muto!
[INFO] [1698663733.004621321] [param_declare]: Hello muto!
[INFO] [1698663735.004378214] [param_declare]: Hello muto!
[INFO] [1698663737.004495248] [param_declare]: Hello muto!
[INFO] [1698663739.005414393] [param_declare]: Hello muto!
[INFO] [1698663741.004875369] [param_declare]: Hello muto!
[INFO] [1698663743.005174937] [param_declare]: Hello muto!
[INFO] [1698663745.005287521] [param_declare]: Hello muto!
[INFO] [1698663747.005402185] [param_declare]: Hello muto!
[INFO] [1698663749.005432493] [param_declare]: Hello robot!
[INFO] [1698663751.005418223] [param_declare]: Hello robot!
[INFO] [1698663753.005454514] [param_declare]: Hello robot!
[INFO] [1698663755.004704535] [param_declare]: Hello robot!
[INFO] [1698663757.005454932] [param_declare]: Hello robot!
[INFO] [1698663759.005436436] [param_declare]: Hello robot!
[INFO] [1698663761.005411059] [param_declare]: Hello robot!
[INFO] [1698663763.005444211] [param_declare]: Hello robot!
[INFO] [1698663765.005400912] [param_declare]: Hello robot!
[INFO] [1698663767.005384622] [param_declare]: Hello robot!
```

```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 param set param_declare robot_name robot
Set parameter successful
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws#
```