

23.ROS2 launch xml.yaml implementation

Continuing from the previous section, this section will explain how to use xml and yaml to write launch files.

1. xml implementation

1.1. Create a new launch file

Create a new file [complex_launch.xml] in the same directory as complex_launch.py and add the following content:

Note: This case will display the little turtle window. Before starting the program, please make sure that the docker GUI display is turned on, otherwise the little turtle window cannot be displayed

```
<launch>

  <!-- args that can be set from the command line or a default will be used -->
  <arg name="background_r" default="0"/>
  <arg name="background_g" default="255"/>
  <arg name="background_b" default="0"/>
  <arg name="chatter_ns" default="my/chatter/ns"/>

  <!-- include another launch file -->
  <include file="$(find-pkg-share
demo_nodes_cpp)/launch/topics/talker_listener.launch.py"/>
  <!-- include another launch file in the chatter_ns namespace-->
  <group>
    <!-- push-ros-namespace to set namespace of included nodes -->
    <push-ros-namespace namespace="$(var chatter_ns)"/>
    <include file="$(find-pkg-share
demo_nodes_cpp)/launch/topics/talker_listener.launch.py"/>
  </group>

  <!-- start a turtlesim_node in the turtlesim1 namespace -->
  <node pkg="turtlesim" exec="turtlesim_node" name="sim"
namespace="turtlesim1"/>
  <!-- start another turtlesim_node in the turtlesim2 namespace
and use args to set parameters -->
  <node pkg="turtlesim" exec="turtlesim_node" name="sim"
namespace="turtlesim2">
    <param name="background_r" value="$(var background_r)"/>
    <param name="background_g" value="$(var background_g)"/>
    <param name="background_b" value="$(var background_b)"/>
  </node>

  <!-- perform remap so both turtles listen to the same command topic -->
  <node pkg="turtlesim" exec="mimic" name="mimic">
    <remap from="/input/pose" to="/turtlesim1/turtle1/pose"/>
  </node>
</launch>
```

```

    <remap from="/output/cmd_vel" to="/turtlesim2/turtle1/cmd_vel"/>
  </node>
</launch>

```

```

1 <launch>
2
3 <!-- args that can be set from the command line or a default will be used -->
4 <arg name="background_r" default="0"/>
5 <arg name="background_g" default="255"/>
6 <arg name="background_b" default="0"/>
7 <arg name="chatter_ns" default="my/chatter/ns"/>
8
9 <!-- include another launch file -->
10 <include file="$(find-pkg-share demo_nodes_cpp)/launch/topics/talker_listener.launch.py"/>
11 <!-- include another launch file in the chatter_ns namespace -->
12 <group>
13   <!-- push-ros-namespace to set namespace of included nodes -->
14   <push-ros-namespace namespace="$(var chatter_ns)"/>
15   <include file="$(find-pkg-share demo_nodes_cpp)/launch/topics/talker_listener.launch.py"/>
16 </group>
17
18 <!-- start a turtlesim_node in the turtlesim1 namespace -->
19 <node pkg="turtlesim" exec="turtlesim_node" name="sim" namespace="turtlesim1"/>
20 <!-- start another turtlesim node in the turtlesim2 namespace
21    and use args to set parameters -->
22 <node pkg="turtlesim" exec="turtlesim_node" name="sim" namespace="turtlesim2">
23   <param name="background_r" value="$(var background_r)"/>
24   <param name="background_g" value="$(var background_g)"/>
25   <param name="background_b" value="$(var background_b)"/>
26 </node>
27 <!-- perform remap so both turtles listen to the same command topic -->
28 <node pkg="turtlesim" exec="mimic" name="mimic">
29   <remap from="/input/pose" to="/turtlesim1/turtle1/pose"/>
30   <remap from="/output/cmd_vel" to="/turtlesim2/turtle1/cmd_vel"/>
31 </node>
32 </launch>

```

1.2. xml configuration

```

1 from setuptools import setup
2 import os
3 from glob import glob
4 package_name = 'pkg_topic'
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, 'launch'), glob(os.path.join('launch', '*launch.xml'))),
16    ],
17    install_requires=['setuptools'],
18    zip_safe=True,
19    maintainer='root',
20    maintainer_email='1461190907@qq.com',
21    description='TODO: Package description',
22    license='TODO: License declaration',
23    tests_require=['pytest'],
24    entry_points={
25        'console_scripts': [
26            'publisher_demo = pkg_topic.publisher_demo:main',
27            'subscriber_demo = pkg_topic.subscriber_demo:main'
28        ],
29    },
30)

```

1.3. Compile workspace

```

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

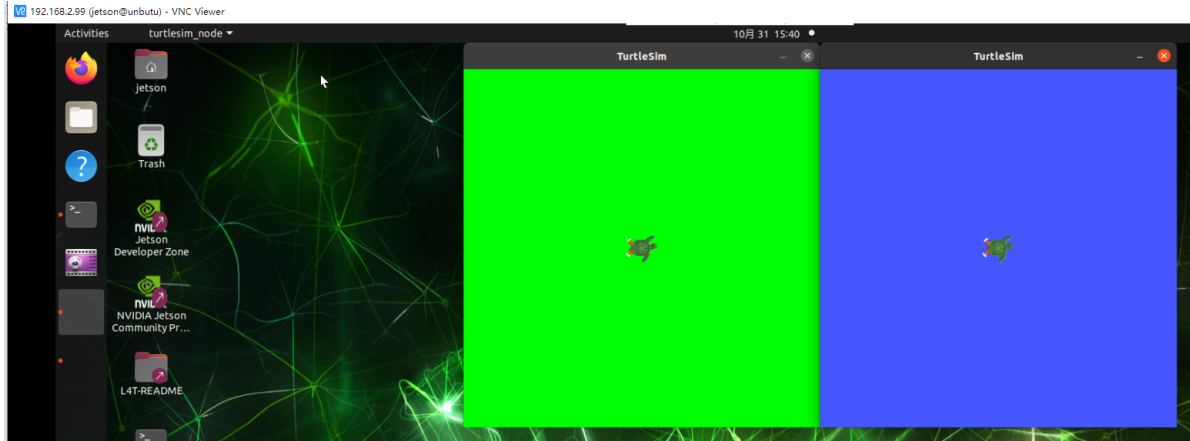
```

1.4. Run the program

Terminal input:

```
ros2 launch pkg_topic complex_launch.xml
```

Two little turtles will be displayed on the host's vnc



2. Yaml implementation

2.1. Create a new launch file

Create a new file [complex_launch.yaml] in the same directory as complex_launch.py and add the following content:

Note: This case will display the little turtle window. Before starting the program, please make sure that the docker GUI display is turned on, otherwise the little turtle window cannot be displayed

```
launch:

# args that can be set from the command line or a default will be used
- arg:
  name: "background_r"
  default: "0"
- arg:
  name: "background_g"
  default: "255"
- arg:
  name: "background_b"
  default: "0"
- arg:
  name: "chatter_ns"
  default: "my/chatter/ns"

# include another launch file
- include:
  file: "$(find-pkg-share
demo_nodes_cpp)/launch/topics/talker_listener.launch.py"
```

```

# include another launch file in the chatter_ns namespace
- group:
  - push-ros-namespace:
      namespace: "${var chatter_ns}"
  - include:
      file: "${find-pkg-share
demo_nodes_cpp)/launch/topics/talker_listener.launch.py"

# start a turtlesim_node in the turtlesim1 namespace
- node:
  pkg: "turtlesim"
  exec: "turtlesim_node"
  name: "sim"
  namespace: "turtlesim1"

# start another turtlesim_node in the turtlesim2 namespace and use args to set
parameters
- node:
  pkg: "turtlesim"
  exec: "turtlesim_node"
  name: "sim"
  namespace: "turtlesim2"
  param:
  -
    name: "background_r"
    value: "${var background_r}"
  -
    name: "background_g"
    value: "${var background_g}"
  -
    name: "background_b"
    value: "${var background_b}"

# perform remap so both turtles listen to the same command topic
- node:
  pkg: "turtlesim"
  exec: "mimic"
  name: "mimic"
  remap:
  -
    from: "/input/pose"
    to: "/turtlesim1/turtle1/pose"
  -
    from: "/output/cmd_vel"
    to: "/turtlesim2/turtle1/cmd_vel"

```

The screenshot shows a ROS2 workspace editor with a file explorer on the left and a code editor on the right. The file explorer shows the directory structure of the 'yahboomcar_ws' workspace, including packages like 'pkg_topic' and 'launch'. The code editor displays the 'complex_launch.yaml' file, which is a launch file for a ROS2 node. The file contains arguments for background colors, includes for other launch files, and a node definition for 'turtlesim' in the 'turtlesim1' namespace.

```
1 launch:
2
3 # args that can be set from the command line or a default will be used
4 - arg:
5   name: "background_r"
6   default: "0"
7 - arg:
8   name: "background_g"
9   default: "255"
10 - arg:
11   name: "background_b"
12   default: "0"
13 - arg:
14   name: "chatter_ns"
15   default: "my/chatter/ns"
16
17 # include another launch file
18 - include:
19   file: "$(find-pkg-share demo_nodes_cpp)/launch/topics/talker_listener.launch.py"
20
21 # include another launch file in the chatter_ns namespace
22 - group:
23   - push-ros-namespace:
24     namespace: "$(var chatter_ns)"
25   - include:
26     file: "$(find-pkg-share demo_nodes_cpp)/launch/topics/talker_listener.launch.py"
27
28 # start a turtlesim_node in the turtlesim1 namespace
29 - node:
30   pkg: "turtlesim"
31   exec: "turtlesim_node"
32   name: "sim"
33   namespace: "turtlesim1"
```

2.2. xml configuration

The screenshot shows a ROS2 workspace editor with a file explorer on the left and a code editor on the right. The file explorer shows the directory structure of the 'yahboomcar_ws' workspace, including packages like 'pkg_topic' and 'launch'. The code editor displays the 'setup.py' file, which is a Python script used to configure the package. The file contains metadata such as name, version, packages, data files, and entry points.

```
1 from setuptools import setup
2 import os
3 from glob import glob
4 package_name = 'pkg_topic'
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, 'launch'), glob(os.path.join('launch', '*launch.xml'))),
16        (os.path.join('share', package_name, 'launch'), glob(os.path.join('launch', '*launch.yaml'))),
17    ],
18    install_requires=['setuptools'],
19    zip_safe=True,
20    maintainer='root',
21    maintainer_email='1461190907@qq.com',
22    description='TODO: Package description',
23    license='TODO: License declaration',
24    tests_require=['pytest'],
25    entry_points={
26        'console_scripts': [
27            'publisher_demo = pkg_topic.publisher_demo:main',
28            'subscriber_demo = pkg_topic.subscriber_demo:main'
29        ],
30    },
31)
```

2.3. Compile workspace

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

2.4. Run the program

Terminal input:

```
ros2 launch pkg_topic complex_launch.yaml
```

Two little turtles will be displayed on the host's vnc

