ROS 2 Basic Concepts

Estimated time to completion: **15 minutes**

2.10 What is a Launch File?

Launch files are scripts that allow you to start **multiple entry points simultaneously**. These entry points can be:

- Within the same script
- In different scripts within the same ROS 2 package
- Across multiple packages, launching different entry points from entirely separate packages

Launch files help streamline the execution of complex ROS 2 systems, making it easier to manage multiple nodes and processes efficiently.

To practice this concept, we'll be working on a new mission!

🚀 MISSION: Start Multiple Rover Heartbeats with a Single Script

In this mission, you'll learn how to use launch files to start multiple instances of the Rover's heartbeat program with a single command. This will help you understand how to efficiently manage multiple ROS 2 nodes within a system.

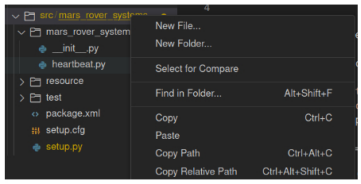
Let's get started!

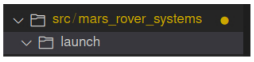
- **REQUIREMENT: You must have completed Exercise 2.2** -

- To use `launch` files, we need the following:
  - A `launch` folder inside the ROS2 package.
  - A `launch` file inside the `launch` folder.
  - Ensure the `launch` folder is included in the `setup.py` so it can be found.
- The goal is to create a `launch` file that can start **TWO nodes at the same time** for two different rovers.

STEP 1: Create the `launch` folder:

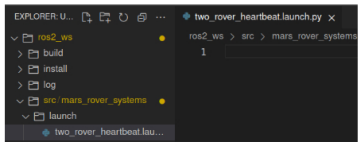
- Go to the IDE and create a folder inside the package `mars_rover_systems`:





STEP 2: Create the `launch` file:

- Inside the `launch` folder, create a new file named `two_rover_heartbeat.launch.py`



`two_rover_heartbeat.launch.py`

In [ ]:

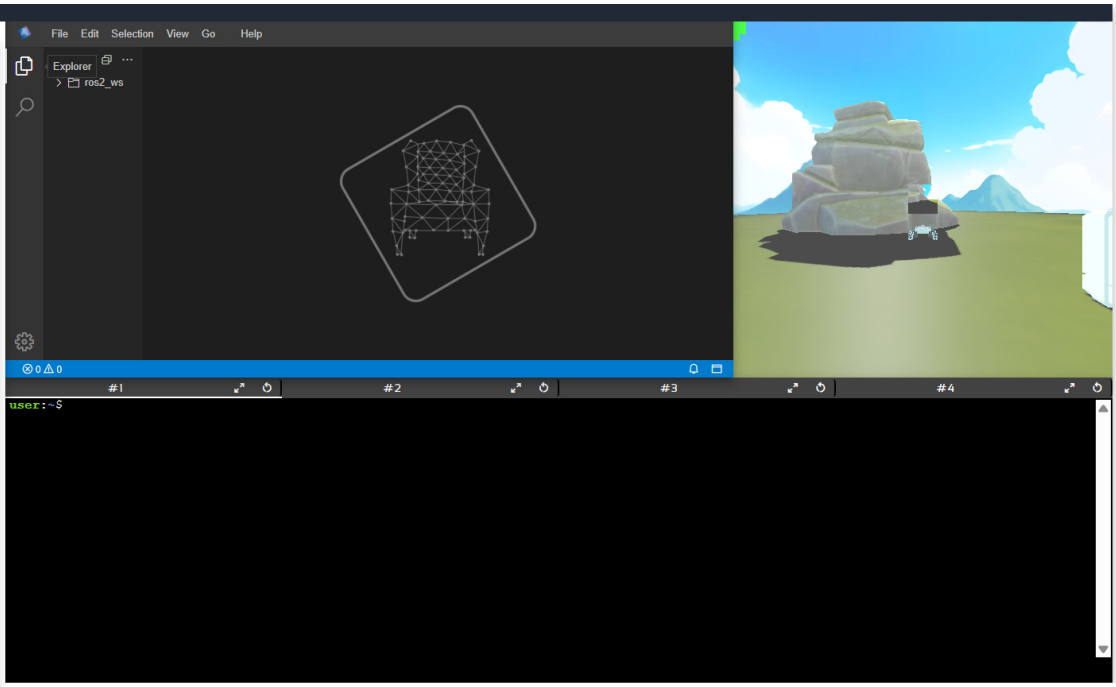
```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='mars_rover_systems',
            executable='heartbeat_executable',
            output='screen'),
        Node(
            package='mars_rover_systems',
            executable='heartbeat_executable2',
            output='screen')
    ])
```

Okay. As you can see, the structure of the `launch` file is quite simple. First, you import some modules from the `launch` and `launch_ros` packages.

In [ ]:

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

Next, define a function that returns a `LaunchDescription` object.



In [ ]:

```
def generate_launch_description():
    return LaunchDescription([
        Node(
            package='mars_rover_systems',
            executable='heartbeat_executable',
            output='screen'),
        Node(
            package='mars_rover_systems',
            executable='heartbeat_executable2',
            output='screen')
    ])
```

Within the `LaunchDescription` object, create a node and specify the following parameters:

1. `package=` The name of the package that contains the ROS2 program to execute.
2. `executable=` The name of the Python `entry point` defined in the `setup.py`.
3. `output=` The type of output channel for printing the program's output.

- There are **MANY** other tools available for use with `launch` files, but this is the bare minimum we need right now.

STEP 3: Modify `setup.py` to find the `.launch.py` files:

- We need to configure `setup.py` to ensure that ROS2 can find and access the `.launch.py` files in the `launch` folder.
- To achieve this, we need to modify the `data_files` entry.
- This entry specifies the locations where **ROS2** can find all the files in our package.
- We need to change the default:

In [ ]:

```
data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
```

- To this:

In [ ]:

```
import os
from glob import glob
...

data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('launc
    ],
```

- We added **ONE NEW PATH**:
  - `os.path.join('share', package_name), glob('launch/*.launch.py')`: which specifies the path to our `launch.py` files.
- This approach can be used for any file or folder that you want your **ROS2 packages** to include.

- At the end, your `setup.py` should look like this:

setup.py

In [ ]:

```
from setuptools import find_packages, setup
import os
from glob import glob

package_name = 'mars_rover_systems'

setup(
    name=package_name,
    version='0.0.0',
    packages=find_packages(exclude=['test']),
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('launc
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='user',
    maintainer_email='user@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'heartbeat_executable = mars_rover_systems.he
            'heartbeat_executable2 = mars_rover_systems.h
        ],
    },
)
```

STEP4) Compile and run

- Compile:

Execute in Terminal #1

In [ ]:

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

```
colcon build
```

- Run the `launch` file.
- To do so we will use the command `ros2 launch`.
- The command has the following structure:

In [ ]:

```
ros2 launch <ros2_package_name> <launch_file_name.launch.
```

- If you press the TAB key twice before typing the `launch.py` file name, it should display all the `launch` files inside the package:

In [ ]:

```
ros2 launch <ros2_package_name> [TAB][TAB]
```

- In our case:

▶ Execute in Terminal #1

In [ ]:

```
ros2 launch mars_rover_systems [TAB][TAB]
```

Terminal #1 Output

```
--debug                         --show-all-subproces
ses-output  -p
--launch-prefix                 --show-args
-s
--launch-prefix-filter          --show-arguments
package.xml
--noninteractive                -a
two_rover_heartbeat.launch.py
--print                         -d
--print-description             -n
```

- This will show all the available command options, including `two_rover_heartbeat.launch.py`.

- TIP -

- **And if you type the first letter of the `launch.py` file and press `TAB`, it will autocomplete the rest of the file name.**

- END TIP -

▶ Execute in Terminal #1

In [ ]:

```
ros2 launch mars_rover_systems two_rover_heartbeat.launch
```

Terminal #1 Output

```
ime(nanoseconds=1724689819484952888, clock_type=ROS_
TIME)
[heartbeat_executable-1] [INFO] [1724689819.49533284
3] [mars_rover_1]: mars_rover_1 is alive...Time(nano
seconds=1724689819494648951, clock_type=ROS_TIME)
[heartbeat_executable2-2] [INFO] [1724689820.4855944
05] [mars_rover_2]: mars_rover_2 is alive...Time(nan
oseconds=1724689820484898191, clock_type=ROS_TIME)
```

- Both `entry points` are being executed at the same time.
- We can also use the `ros2 node list` command to verify that **BOTH NODES** are running:

▶ Execute in Terminal #2

In [ ]:

```
ros2 node list
```

Terminal #2 Output

```
/clock_bridge
/cmd_vel_bridge
/fake_joint_state_publisher
/laser_bridge
/mars_rover_1
/mars_rover_2
/odom_bridge
/rgb_camera_bridge
/robot_state_publisher
```

- Notes -

`Launch` files are useful because they offer a convenient way to start multiple nodes with a single file, and allow for node configuration, such as setting parameters. For more details on `launch` files and work with parameters in ROS2 in our Intermediate ROS2 course.

- End of Notes -

- Exercise 2.3 -

Create a new script that simulates the temperature monitoring of our mars rover.

- Create a new script in the `mars_rover_systems` package, a new script named `temperature_monitor.py`.
- The `entry point` name should be `temperaturemonitor_executable`.
- Create a **custom ROS2 node class**.
- Instead of using the `main()` method name, use `start_monitor()` to contain all the node initialization and `spin()`.
- Create a new `launch` file that starts the `entry points` of `heartbeat_executable` and `temperaturemonitor_executable`, starting both **heartbeat system** and the **temperature monitor system**. The name should

be `start_mars_rover_systems.launch.py`.
- It has to simulate temperature fluctuations using the python module method `random.uniform(20.0, 100.0)`.
- It should print on the screen the temperature of the mars rover every **second**.
- If the **temperature** exceeds a threshold of `70.0 ºC`, which is considered high temperature, has to issue a warning message.
- `WARNING` **MESSAGES** in ROS2 are issued by changing the `info`:

In [ ]:
```
self.get_logger().info("")
```

To `warn`:

In [ ]:
```
self.get_logger().warn("")
```

- End of Exercise 2.3 -

- TIP -

- To enable colors to appear in the terminal and see the `WARNING` messages, add the `emulate_TTY` option to the Node `launch`:

In [ ]:
```
def generate_launch_description():
    return LaunchDescription([
        Node(
            package='mars_rover_systems',
            executable='heartbeat_executable',
            output='screen',
            emulate_tty=True),
```

- END TIP -

- Expected result for Exercise 2.3 -

- Compile:

▶ Execute in Terminal #1

In [ ]:
```
cd ~/ros2_ws
```

In [ ]:
```
colcon build
```

In [ ]:
```
source ~/ros2_ws/install/setup.bash
```

- Start:

▶ Execute in Terminal #1

In [ ]:
```
ros2 launch mars_rover_systems start_mars_rover_systems.l
```

Terminal #1 Output

```
[temperaturemonitor_executable-2] [INFO] [172469203
0.622757554] [temperature_monitor]: Current temperat
ure: 94.97°C
```

```
[temperaturemonitor_executable-2] [WARN] [172469203
0.623391831] [temperature_monitor]: Warning: High te
mperature detected! 94.97°C
```

```
[heartbeat_executable-1] [INFO] [1724692030.62634458
2] [mars_rover_1]: mars_rover_1 is alive...Time(nano
seconds=1724692030625562354, clock_type=ROS_TIME)
[temperaturemonitor_executable-2] [INFO] [172469203
1.622914526] [temperature_monitor]: Current temperat
ure: 27.89°C
[heartbeat_executable-1] [INFO] [1724692031.62621542
1] [mars_rover_1]: mars_rover_1 is alive...Time(nano
seconds=1724692031625562309, clock_type=ROS_TIME)
```

- See that when the temeprature is `94.97°C` > `70 ºC` it issues a `warning` message
- We also see the 'mars_rover_1 heartbeat.

▶ Execute in Terminal #2

In [ ]:
```
ros2 node list
```

Terminal #2 Output

```
/clock_bridge
/cmd_vel_bridge
/fake_joint_state_publisher
/laser_bridge
/mars_rover_1
/odom_bridge
/rgb_camera_bridge
/robot_state_publisher
/temperature_monitor
```

- You should have **TWO new nodes**: `/temperature_monitor` and `/mars_rover_1`.

- End Expected result for Exercise 2.3 -

- RECOMMENDATION -

PLEASE TRY TO DO THE EXERCISE BEFORE LOOKING AT

SOLUTION

- END RECOMMENDATION -

- Solution for Exercise 2.3 -

temperature_monitor.py

In [ ]:

```
import rclpy
from rclpy.node import Node
import random

class TemperatureMonitorNode(Node):
    def __init__(self):
        super().__init__('temperature_monitor')
        self.temperature_threshold = 70.0  # Set the temp
        self.get_logger().info('Temperature Monitor Node

        # Create a timer that triggers every second (1 Hz
        self.timer = self.create_timer(1.0, self.monitor_

    def get_temperature(self):
        # Simulate getting a temperature reading (in real
        temperature = random.uniform(20.0, 100.0)
        return temperature

    def monitor_temperature_callback(self):
        current_temperature = self.get_temperature()
        self.get_logger().info(f'Current temperature: {cu

        if current_temperature > self.temperature_thresho
            self.get_logger().warn(f'Warning: High temper

def start_monitor(args=None):
    rclpy.init(args=args)
    node = TemperatureMonitorNode()

    try:
        rclpy.spin(node)  # Keep the node running, proces
    except KeyboardInterrupt:
        pass

    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    start_monitor()
```

start_mars_rover_systems.launch.py

In [ ]:

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

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='mars_rover_systems',
            executable='heartbeat_executable',
            output='screen',
            emulate_tty=True),
        Node(
            package='mars_rover_systems',
            executable='temperaturemonitor_executable',
            output='screen',
            emulate_tty=True)
    ])
```

setup.py

In [ ]:

```
from setuptools import find_packages, setup
import os
from glob import glob

package_name = 'mars_rover_systems'

setup(
    name=package_name,
    version='0.0.0',
    packages=find_packages(exclude=['test']),
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('launc
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='user',
    maintainer_email='user@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
```

```
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'heartbeat_executable = mars_rover_systems.he
            'heartbeat_executable2 = mars_rover_systems.h
            'temperaturemonitor_executable = mars_rover_s
        ],
    },
)
```

- End of Solution for Exercise 2.3 -

24/09/2024