ROS 2 Basic Concepts

Estimated time to completion: **7 minutes**

2.4 Create a Package

Now that you understand what a **ROS 2 package** is, it's time to create your own!

Up until now, you've been examining the structure of an existing ROS 2 package. Now, it's time to put your knowledge into practice. We need a place to store all the programs related to checking the systems of the Mars Rover, and this is a perfect use case for a ROS 2 package. In ROS, it's common to group all programs related to a specific function within a single package.

Let's get started by creating a new ROS 2 package, named `mars_rover_systems`. This will be the foundation for managing and organizing all the systems of the Mars Rover.

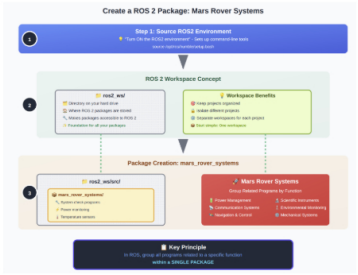
Before diving into the practical example, let's review some key points related to ROS 2 packages:

- When creating packages, you need to work within a specific workspace called a **ROS 2 workspace**. This workspace is a directory on your hard drive where your ROS 2 packages are stored and made accessible to ROS 2. Typically, the directory for a ROS 2 workspace is named `ros2_ws`.
- For different ROS 2 projects, you may choose to create separate workspaces to keep each project isolated from others.

To keep things simple, let's start with a single workspace.

- Example 2.2 -

- First, source the ROS2 environment in your Terminal to use the ROS2 command-line tools.
- Sourcing sets up ROS2 environment so that you can use its command-line tools in that terminal.
- Think of it as **turning ON the ROS2 environment**.



Execute in Terminal #1

```
In [ ]:
source /opt/ros/humble/setup.bash
```

Now, go to the `ros2_ws` in your Terminal #1

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

```
In [ ]:
pwd
```

This will give you the following as output:

Terminal #1 Output

```
/home/user/ros2_ws
```

Inside this workspace, there is a directory called `src`. This folder contains all the packages created. Every time you want to create a package, you have to be in this directory `ros2_ws/src`. Type into your Terminal the following command:

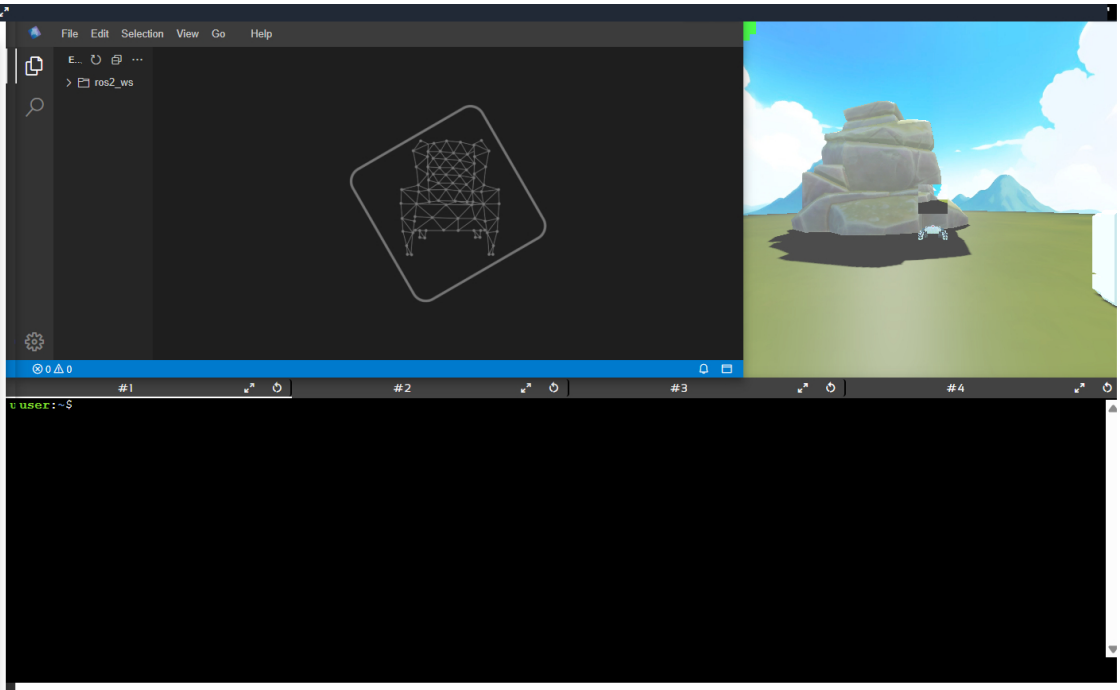
Execute in Terminal #1

```
In [ ]:
cd src
```

- Now let's create the package `mars_rover_systems` in the path `/home/user/ros2_ws/src`
- **AGAIN**: **ALWAYS place all the ROS2 packages inside the workspace's `src` folder**. Otherwise you will have issues when compiling.

Execute in Terminal #1

```
In [ ]:
ros2 pkg create --build-type ament_python mars_rover_syst
```



Something similar to the message below will appear in your terminal:

Terminal #1 Output

```
going to create a new package
package name: mars_rover_systems
destination directory: /home/user/ros2_ws
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['user <>']
licenses: ['TODO: License declaration']
build type: ament_python
dependencies: ['rclpy']
creating folder ./mars_rover_systems
creating ./mars_rover_systems/package.xml
creating source folder
creating folder ./mars_rover_systems/mars_rover_systems
creating ./mars_rover_systems/setup.py
creating ./mars_rover_systems/setup.cfg
creating folder ./mars_rover_systems/resource
creating ./mars_rover_systems/resource/mars_rover_systems
creating ./mars_rover_systems/mars_rover_systems/__init__.py
creating folder ./mars_rover_systems/test
creating ./mars_rover_systems/test/test_copyright.py
creating ./mars_rover_systems/test/test_flake8.py
creating ./mars_rover_systems/test/test_pep257.py

[WARNING]: Unknown license 'TODO: License declaration'.  This has been set in the package.xml, but no LICENSE file has been created.
It is recommended to use one of the ament license identifiers:
Apache-2.0
BSL-1.0
BSD-2.0
BSD-2-Clause
BSD-3-Clause
GPL-3.0-only
LGPL-3.0-only
MIT
MIT-0
```

Inside your `src` directory, this command creates a new package with files. You will check this later. For now, see how this command is built:

In [ ]:

```
ros2 pkg create --build-type ament_python <package_name>
```

The `<package_name>` is the name of the package you want to create, and the `<package_dependency_X>` are the names of other ROS2 packages that your package depends on.

Note also that we are specifying `ament_python` as the `build type`. This indicates that we are creating a Python package.

Building your package after creating it is a good practice. It allows you to quickly verify whether the listed dependencies can be resolved and helps identify any errors in the provided information.

Execute in Terminal #1

In [ ]:

```
cd ~/ros2_ws/
```

In [ ]:

```
colcon build
```

Terminal #1 Output

```
Finished <<< mars_systems [1.94s]

Summary: 1 package finished [2.76s]
```

- Make it a habit to **source** the `setup.bash` file from the `install` folder so that ROS 2 can locate the packages in your workspace.
- Think of it as **turning on a switch for your ROS 2 packages** inside `ros2_ws`, enabling ROS 2 to recognize and execute the programs within them.

Execute in Terminal #1

In [ ]:

```
source install/setup.bash
```

- End of Example 2.2 -

- Example 2.3 -

To confirm that your package has been created successfully, use ROS commands related to packages. For example, type the following:

Execute in Terminal #1

In [ ]:

```
ros2 pkg list
```

In [ ]:

```
ros2 pkg list | grep mars_rover_systems
```

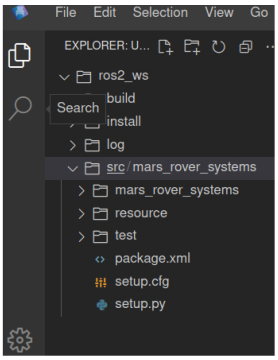
Terminal #1 Output

```
mars_rover_systems
```

`ros2 pkg list:` Gives you a list of all the ROS packages in your system.

`ros2 pkg list | grep mars_rover_systems` : Filters the list to show only the package named **`mars_rover_systems`**.

You can also view the package created and its contents by opening it in the IDE:



If the above commands do not show anything in the terminal, do not panic. this is normal. As with other coding tasks, you need to compile your code to generate executables, We will cover how to do this in the next section.

- End of Example 2.3 -

- Notes -

Packages are organized within workspaces. Each workspace can contain as many packages as you need. For this course, your workspace is named `ros2_ws`. The overall structure could look as follows:

```
ros2_ws/
    src/
        mars_rover_systems/
            package.xml
            setup.py
            ...
        mars_rover_anotherpackage/
            package.xml
            setup.py
            ...
        mars_rover_anotherpackage_2/
            package.xml
            setup.py
            ...
```

- End of Notes -

24/09/2024