Configuring environment

Goal: This tutorial will show you how to prepare your ROS 2 environment.

Tutorial level: Beginner

Time: 5 minutes

Contents

Background

ROS 2 relies on the notion of combining workspaces using the shell environment. “Workspace” is a ROS term for the location on your system where you’re developing with ROS 2. The core ROS 2 workspace is called the underlay. Subsequent local workspaces are called overlays. When developing with ROS 2, you will typically have several workspaces active concurrently.

Combining workspaces makes developing against different versions of ROS 2, or against different sets of packages, easier. It also allows the installation of several ROS 2 distributions (or “distros”, e.g. Dashing and Eloquent) on the same computer and switching between them.

This is accomplished by sourcing setup files every time you open a new shell, or by adding the source command to your shell startup script once. Without sourcing the setup files, you won’t be able to access ROS 2 commands, or find or use ROS 2 packages. In other words, you won’t be able to use ROS 2.

Prerequisites

Before starting these tutorials, install ROS 2 by following the instructions on the ROS 2 Installation page.

The commands used in this tutorial assume you followed the binary packages installation guide for your operating system (Debian packages for Linux). You can still follow along if you built from source, but the path to your setup files will likely be different. You also won't be able to use the `sudo apt install ros-<distro>-<package>` command (used frequently in the beginner level tutorials) if you install from source.

If you are using Linux or macOS, but are not already familiar with the shell, this tutorial will help.

Tasks

1 Source the setup files

You will need to run this command on every new shell you open to have access to the ROS 2 commands, like so:

Linux | macOS | Windows

```
# Replace ".bash" with your shell if you're not using bash
# Possible values are: setup.bash, setup.sh, setup.zsh
source /opt/ros/humble/setup.bash
```

Note

The exact command depends on where you installed ROS 2. If you're having problems, ensure the file path leads to your installation.

2 Add sourcing to your shell startup script

If you don't want to have to source the setup file every time you open a new shell (skipping task 1), then you can add the command to your shell startup script:

Linux | macOS | Windows

```
echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
```

To undo this, locate your system's shell startup script and remove the appended source command.

3 Check environment variables

Sourcing ROS 2 setup files will set several environment variables necessary for operating ROS 2. If you ever have problems finding or using your ROS 2 packages, make sure that your environment is properly set up using the following command:

Linux | macOS | Windows

```
printenv | grep -i ROS
```

Check that variables like `ROS_DISTRO` and `ROS_VERSION` are set.

```
ROS_VERSION=2
ROS_PYTHON_VERSION=3
ROS_DISTRO=humble
```

If the environment variables are not set correctly, return to the ROS 2 package installation section of the installation guide you followed. If you need more specific help (because environment setup files can come from different places), you can get answers from the community.

3.1 The `ROS_DOMAIN_ID` variable

See the domain ID article for details on ROS domain IDs.

Once you have determined a unique integer for your group of ROS 2 nodes, you can set the environment variable with the following command:

Linux | macOS | Windows

```
export ROS_DOMAIN_ID=<your_domain_id>
```

To maintain this setting between shell sessions, you can add the command to your shell startup script:

```
echo "export ROS_DOMAIN_ID=<your_domain_id>" >> ~/.bashrc
```

3.2 The `ROS_LOCALHOST_ONLY` variable

By default, ROS 2 communication is not limited to localhost. `ROS_LOCALHOST_ONLY` environment variable allows you to limit ROS 2 communication to localhost only. This means your ROS 2 system, and its topics, services, and actions will not be visible to other computers on the local network. Using `ROS_LOCALHOST_ONLY` is helpful in certain settings, such as classrooms, where multiple robots may publish to the same topic causing strange behaviors. You can set the environment variable with the following command:

Linux | macOS | Windows

```
export ROS_LOCALHOST_ONLY=1
```

To maintain this setting between shell sessions, you can add the command to your shell startup script:

```
echo "export ROS_LOCALHOST_ONLY=1" >> ~/.bashrc
```

Summary

The ROS 2 development environment needs to be correctly configured before use. This can be done in two ways: either sourcing the setup files in every new shell you open, or adding the source command to your startup script.

If you ever face any problems locating or using packages with ROS 2, the first thing you should do is check your environment variables and ensure they are set to the version and distro you intended.

Next steps

Now that you have a working ROS 2 installation and you know how to source its setup files, you can start learning the ins and outs of ROS 2 with the turtlesim tool.