16. ROS2 Common Command Tools

1. Package Management Tool ros2 pkg

1.1. ros2 pkg create

Function: Creates a package. When creating a package, you must specify the package name, compilation method, dependencies, etc.

Format:

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

In the ros2 command:

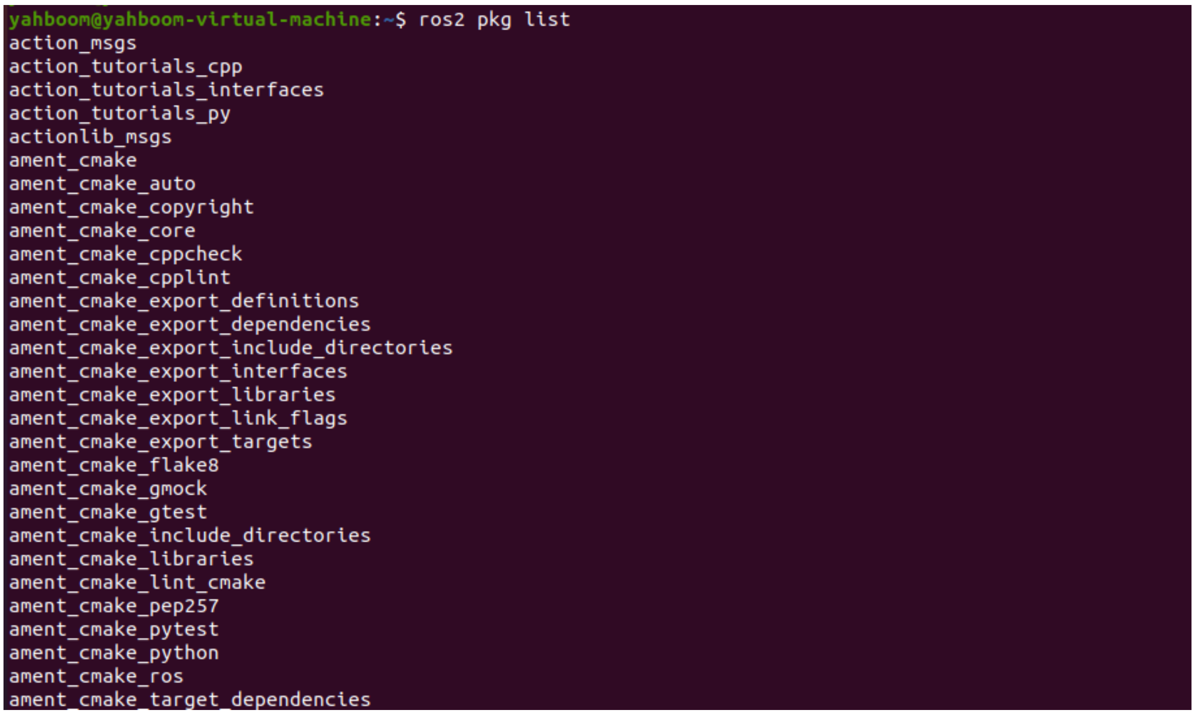
- **pkg**: Indicates the functions associated with the package;
- **create**: Indicates the creation of a package;
- **package_name**: Required: The name of the new package;
- **build-type**: Required: Indicates whether the newly created package is C++ or Python. If using C++ or C, follow ament_cmake; if using Python, follow ament_python;
- **dependencies**: Optional: Indicates the package's dependencies. C++ packages must include rclcpp; Python packages must include rclpy, as well as other required dependencies.

1.2, ros2 pkg list

Function: View the list of packages in the system

Format:

```
ros2 pkg list
```



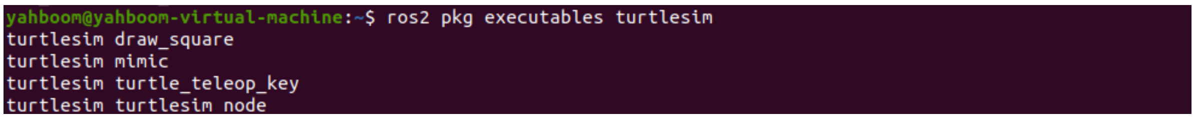
```
yahboom@yahboom-virtual-machine:~$ ros2 pkg list
action_msgs
action_tutorials_cpp
action_tutorials_interfaces
action_tutorials_py
actionlib_msgs
ament_cmake
ament_cmake_auto
ament_cmake_copyright
ament_cmake_core
ament_cmake_cppcheck
ament_cmake_cpplint
ament_cmake_export_definitions
ament_cmake_export_dependencies
ament_cmake_export_include_directories
ament_cmake_export_interfaces
ament_cmake_export_libraries
ament_cmake_export_link_flags
ament_cmake_export_targets
ament_cmake_flake8
ament_cmake_gmock
ament_cmake_gtest
ament_cmake_include_directories
ament_cmake_libraries
ament_cmake_lint_cmake
ament_cmake_pep257
ament_cmake_pytest
ament_cmake_python
ament_cmake_ros
ament_cmake_target_dependencies
```

1.3. ros2 pkg executables

Function: View all executable files in a package

Format:

```
ros2 pkg executables pkg_name
```



```
yahboom@yahboom-virtual-machine:~$ ros2 pkg executables turtlesim
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
```

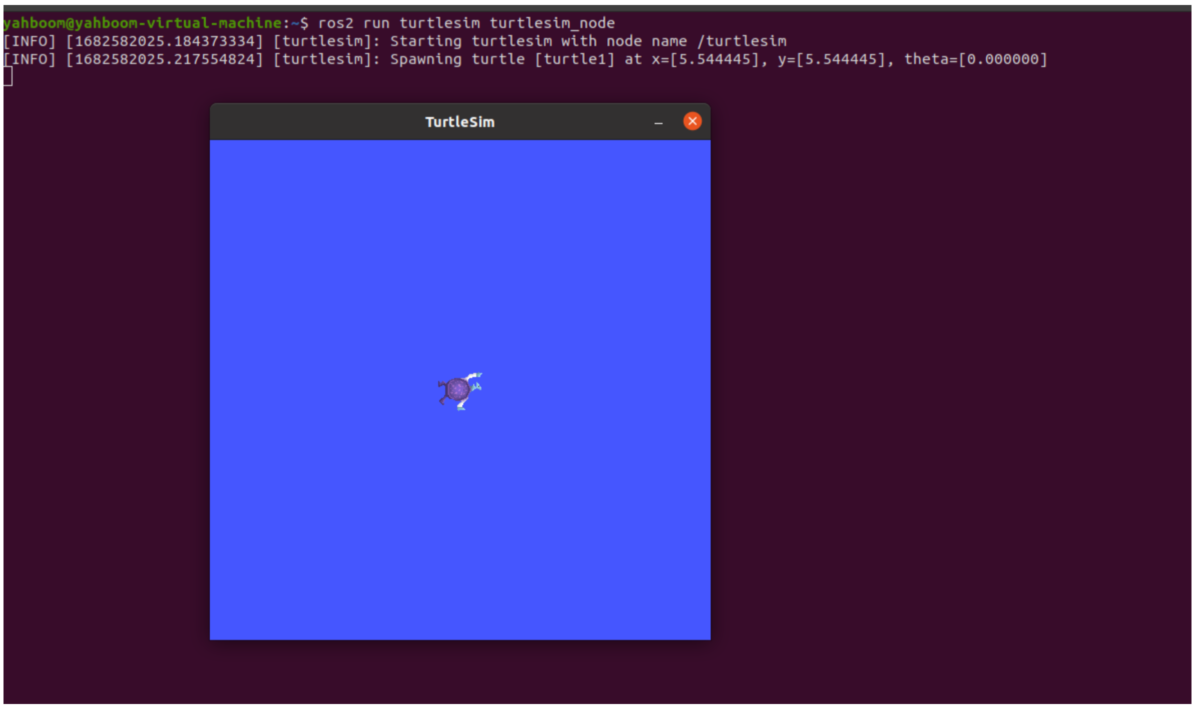
2. Node Run ros2 run

Function: Run the node program in the package

Format:

```
ros2 run pkg_name node_name
```

- pkg_name: Package name
- node_name: The name of the executable program



```
yahboom@yahboom-virtual-machine:~$ ros2 run turtlesim turtlesim_node
[INFO] [1682582025.184373334] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1682582025.217554824] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

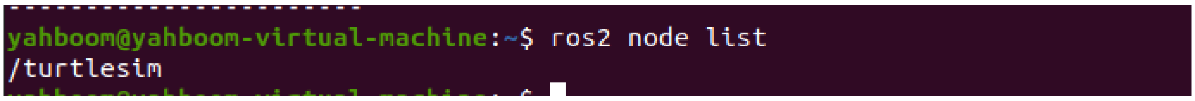
3. Node-Related Tools: ros2 node

3.1. ros2 node list

Function: Lists all node names in the current domain

Format:

```
ros2 node list
```



```
--------------------------
yahboom@yahboom-virtual-machine:~$ ros2 node list
/turtlesim
yahboom@yahboom-virtual-machine:~$
```

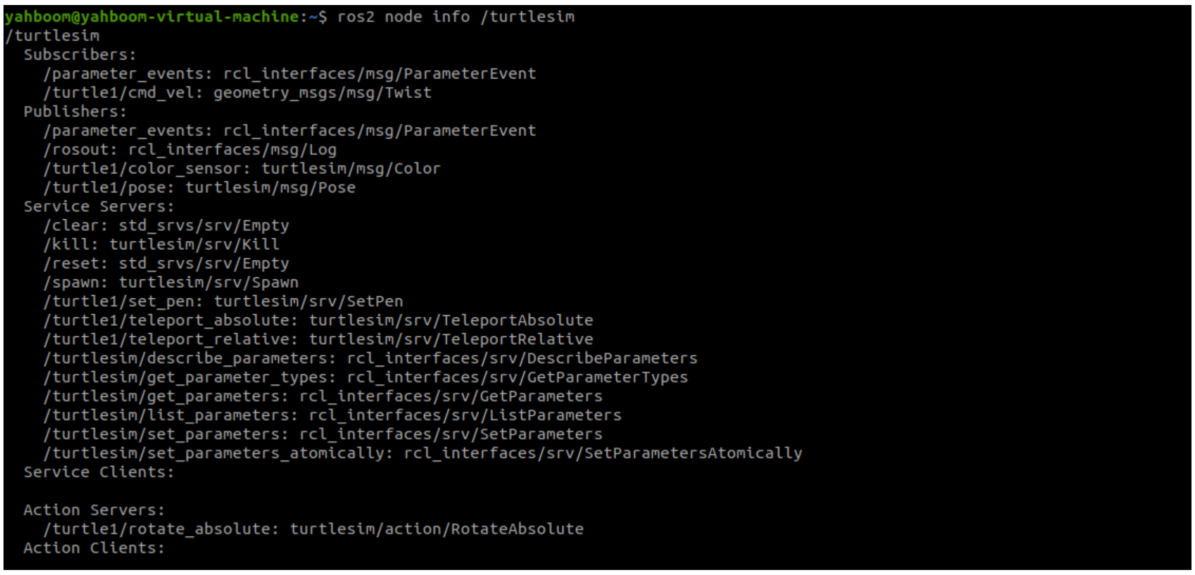
3.2. ros2 node info

Function: View detailed node information, including subscriptions, published messages, enabled services, and actions.

Format:

```
ros2 node info node_name
```

- node_name: The name of the node to be viewed.



```
yahboom@yahboom-virtual-machine:~$ ros2 node info /turtlesim
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:
```

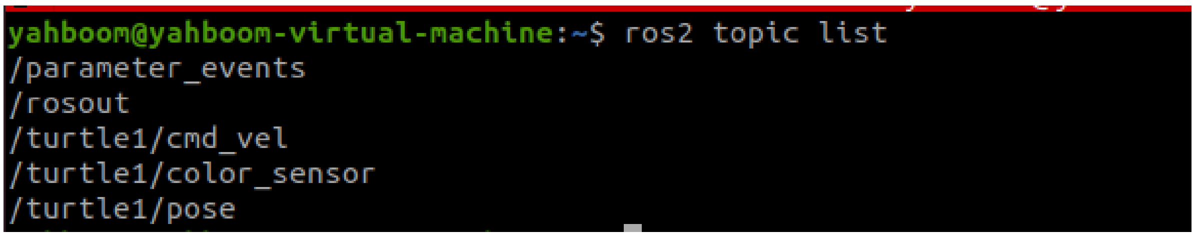
4. Topic-Related Tools: ros2 topic

4.1. ros2 topic list

Function: List all topics in the current domain

Format:

```
ros2 topic list
```



```
yahboom@yahboom-virtual-machine:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

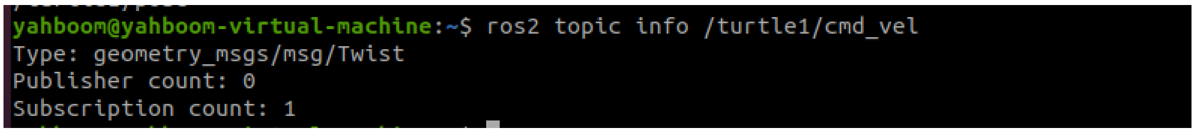
4.2. ros2 topic info

Function: Display topic message type and number of subscribers/publishers

Format:

```
ros2 topic info topic_name
```

- topic_name: The name of the topic to be queried.



```
yahboom@yahboom-virtual-machine:~$ ros2 topic info /turtle1/cmd_vel
Type: geometry_msgs/msg/Twist
Publisher count: 0
Subscription count: 1
```

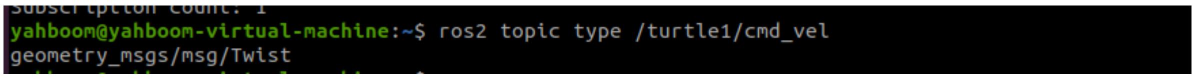
4.3, ros2 topic type

Function: View the message type of a topic

Format:

```
ros2 topic type topic_name
```

- topic_name: The name of the topic type to be queried.



```
yahboom@yahboom-virtual-machine:~$ ros2 topic type /turtle1/cmd_vel
geometry_msgs/msg/Twist
```

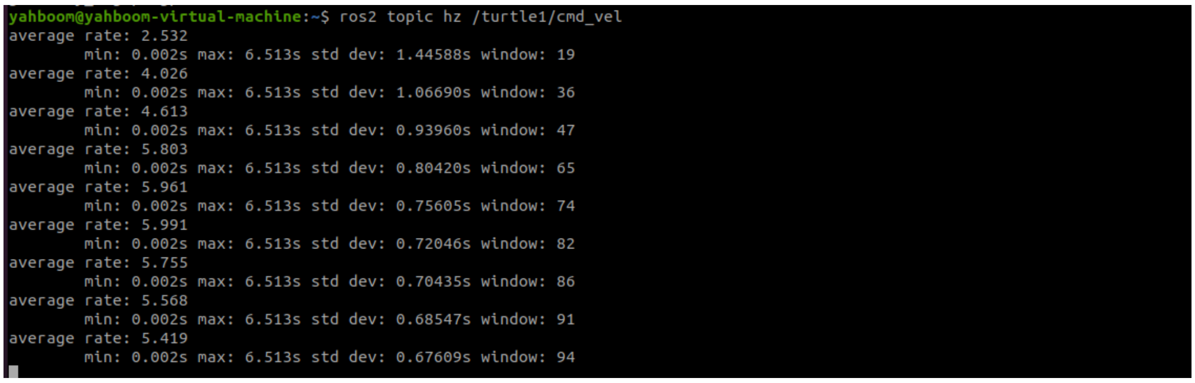
4.4, ros2 topic hz

Function: Display the average publishing frequency of a topic.

Format:

```
ros2 topic hz topic_name
```

- topic_name: The name of the topic whose frequency you want to query.



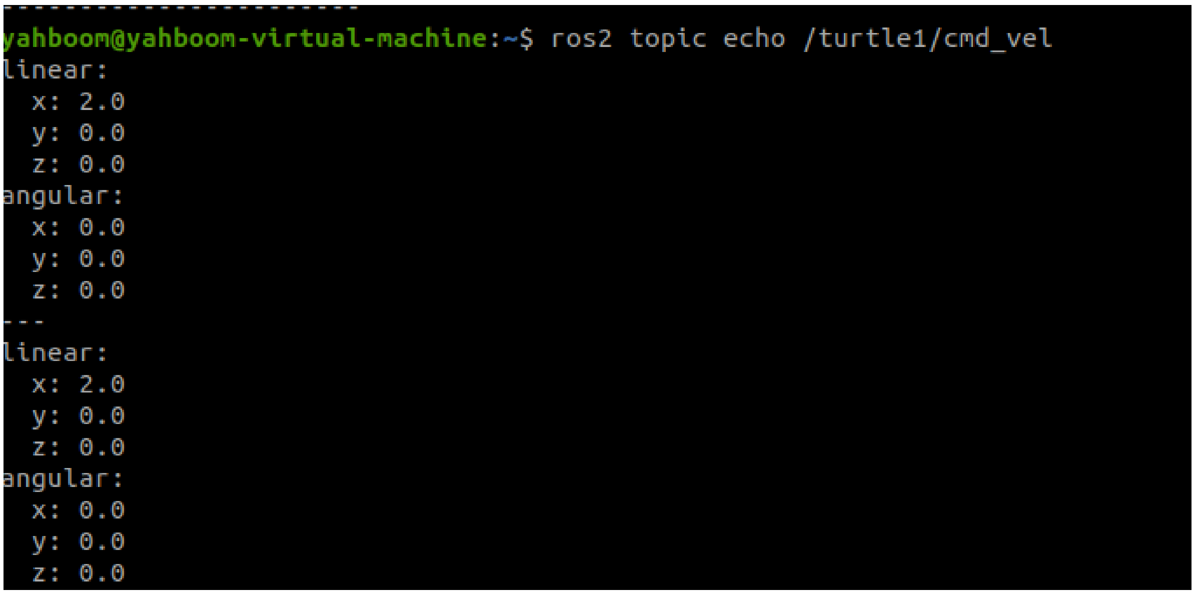
```
yahboom@yahboom-virtual-machine:~$ ros2 topic hz /turtle1/cmd_vel
average rate: 2.532
        min: 0.002s max: 6.513s std dev: 1.44588s window: 19
average rate: 4.026
        min: 0.002s max: 6.513s std dev: 1.06690s window: 36
average rate: 4.613
        min: 0.002s max: 6.513s std dev: 0.93960s window: 47
average rate: 5.803
        min: 0.002s max: 6.513s std dev: 0.80420s window: 65
average rate: 5.961
        min: 0.002s max: 6.513s std dev: 0.75605s window: 74
average rate: 5.991
        min: 0.002s max: 6.513s std dev: 0.72046s window: 82
average rate: 5.755
        min: 0.002s max: 6.513s std dev: 0.70435s window: 86
average rate: 5.568
        min: 0.002s max: 6.513s std dev: 0.68547s window: 91
average rate: 5.419
        min: 0.002s max: 6.513s std dev: 0.67609s window: 94
```

4.5, ros2 topic echo

Function: Print topic messages on the terminal, similar to a subscriber.

Format: ros2 topic echo topic_name

- topic_name: The name of the topic whose messages you want to print.



```
yahboom@yahboom-virtual-machine:~$ ros2 topic echo /turtle1/cmd_vel
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
```

4.5, ros2 topic pub

Function: Publish a message on a specified topic on the terminal.

Format:

```
ros2 topic pub topic_name message_type message_content
```

- topic_name: The name of the topic whose messages you want to publish.
- message_type: The data type of the topic.
- message_content: Message content

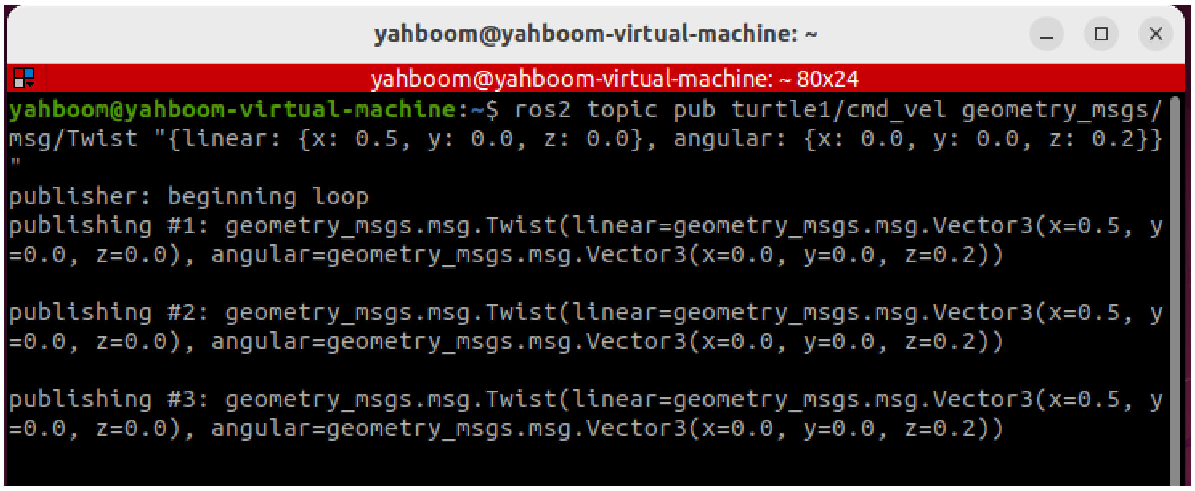
The default is to publish at a 1Hz frequency. The following parameters can be set:

- Parameter -1 to publish only once, ros2 topic pub -1 topic_name message_type message_content
- Parameter -t count to publish count times, ros2 topic pub -t count topic_name message_type message_content
- Parameter -r count to publish at a count Hz frequency, ros2 topic pub -r count topic_name message_type message_content

Example:

- Publish velocity commands via the command line
- Note that there is a space after each colon; otherwise, a format error will be displayed.

```
ros2 topic pub turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.5, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.2}}"
```


yahboom@yahboom-virtual-machine: ~
yahboom@yahboom-virtual-machine: ~ 80x24

```
yahboom@yahboom-virtual-machine:~$ ros2 topic pub turtle1/cmd_vel geometry_msgs/
msg/Twist "{linear: {x: 0.5, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.2}}
"
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y
=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.2))

publishing #2: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y
=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.2))

publishing #3: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.5, y
=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.2))
```

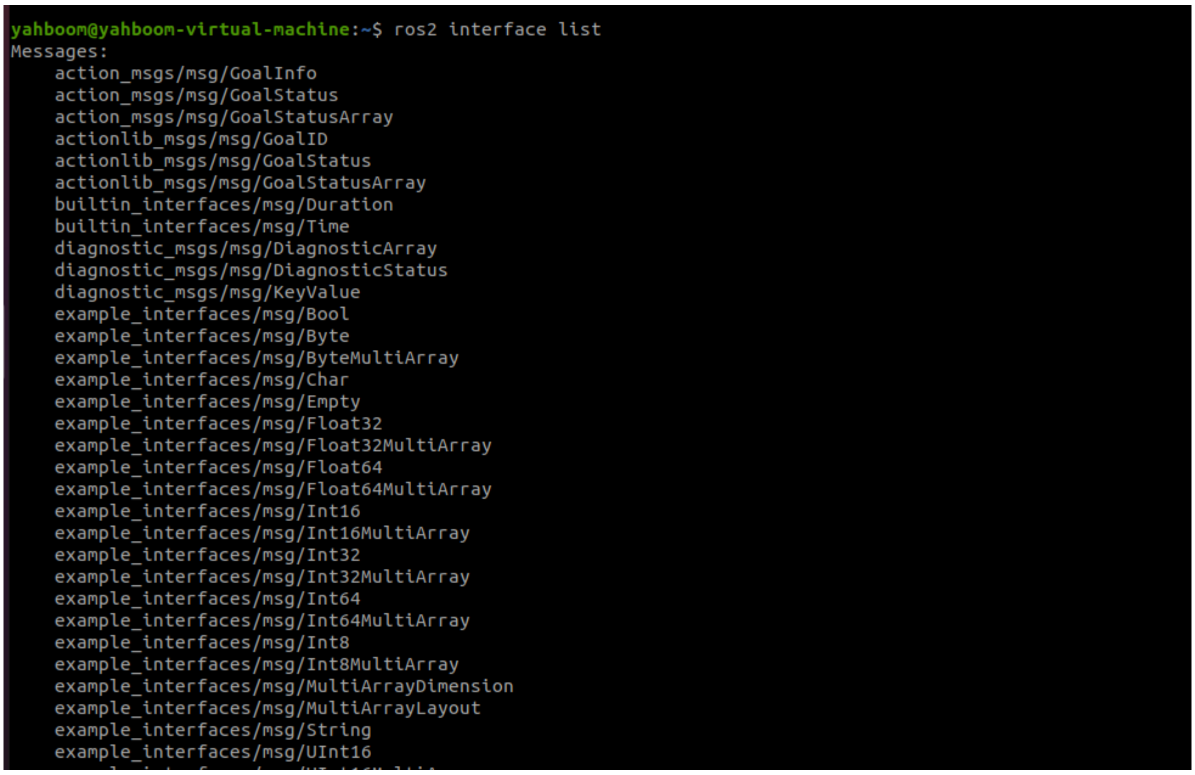
5. Interface-Related Tools: ros2 interface

5.1. ros2 interface list

Function: Lists all interfaces in the current system, including topics, services, and actions.

Format:

```
ros2 interface list
```



```
yahboom@yahboom-virtual-machine:~$ ros2 interface list
Messages:
    action_msgs/msg/GoalInfo
    action_msgs/msg/GoalStatus
    action_msgs/msg/GoalStatusArray
    actionlib_msgs/msg/GoalID
    actionlib_msgs/msg/GoalStatus
    actionlib_msgs/msg/GoalStatusArray
    builtin_interfaces/msg/Duration
    builtin_interfaces/msg/Time
    diagnostic_msgs/msg/DiagnosticArray
    diagnostic_msgs/msg/DiagnosticStatus
    diagnostic_msgs/msg/KeyValue
    example_interfaces/msg/Bool
    example_interfaces/msg/Byte
    example_interfaces/msg/ByteMultiArray
    example_interfaces/msg/Char
    example_interfaces/msg/Empty
    example_interfaces/msg/Float32
    example_interfaces/msg/Float32MultiArray
    example_interfaces/msg/Float64
    example_interfaces/msg/Float64MultiArray
    example_interfaces/msg/Int16
    example_interfaces/msg/Int16MultiArray
    example_interfaces/msg/Int32
    example_interfaces/msg/Int32MultiArray
    example_interfaces/msg/Int64
    example_interfaces/msg/Int64MultiArray
    example_interfaces/msg/Int8
    example_interfaces/msg/Int8MultiArray
    example_interfaces/msg/MultiArrayDimension
    example_interfaces/msg/MultiArrayLayout
    example_interfaces/msg/String
    example_interfaces/msg/UInt16
```

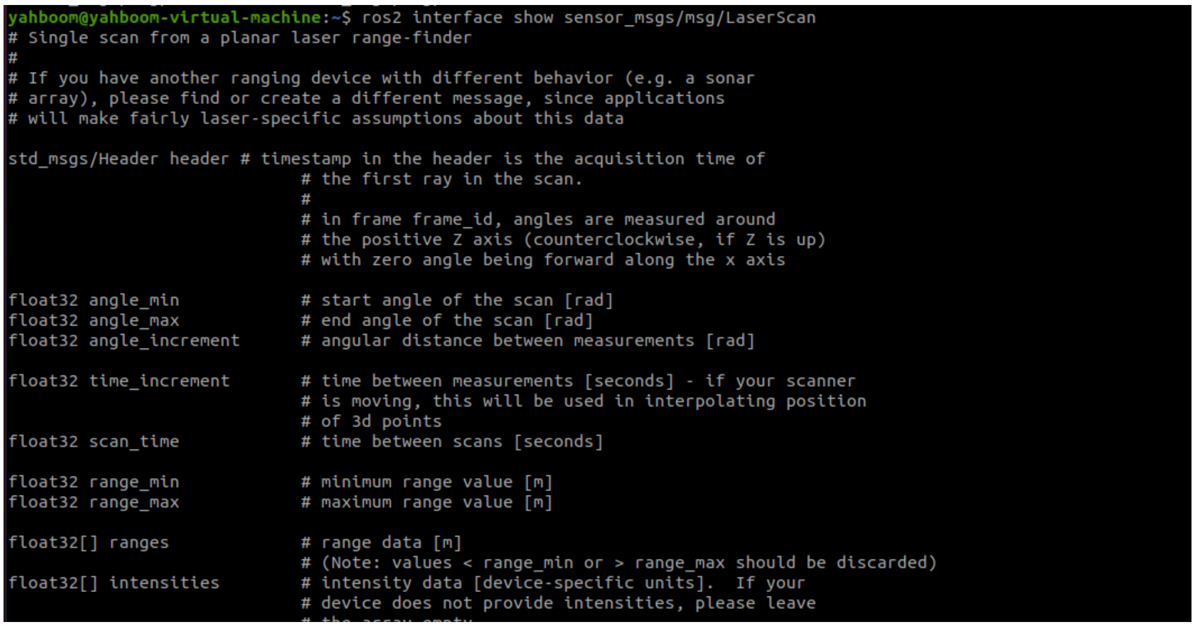
5.2. ros2 interface show

Function: Displays the detailed information of a specified interface

Format:

```
ros2 interface show interface_name
```

- interface_name: The name of the interface to be displayed


```
yahboom@yahboom-virtual-machine:~$ ros2 interface show sensor_msgs/msg/LaserScan
# Single scan from a planar laser range-finder
#
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

std_msgs/Header header # timestamp in the header is the acquisition time of
                             # the first ray in the scan.
                             #
                             # in frame frame_id, angles are measured around
                             # the positive Z axis (counterclockwise, if Z is up)
                             # with zero angle being forward along the x axis

float32 angle_min            # start angle of the scan [rad]
float32 angle_max            # end angle of the scan [rad]
float32 angle_increment      # angular distance between measurements [rad]

float32 time_increment       # time between measurements [seconds] - if your scanner
                             # is moving, this will be used in interpolating position
                             # of 3d points
float32 scan_time            # time between scans [seconds]

float32 range_min            # minimum range value [m]
float32 range_max            # maximum range value [m]

float32[] ranges             # range data [m]
                             # (Note: values < range_min or > range_max should be discarded)
float32[] intensities        # intensity data [device-specific units].  If your
                             # device does not provide intensities, please leave
```

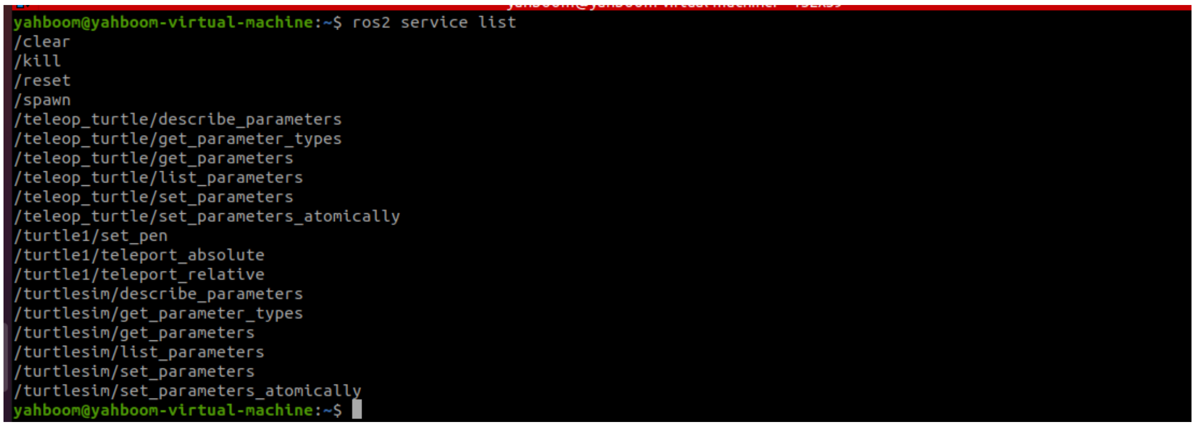
6. Service-Related Tools ros2 service

6.1. ros2 service list

Function: Lists all services in the current domain

Format:

```
ros2 interface show interface_name
```


```
yahboom@yahboom-virtual-machine:~$ ros2 service list
/clear
/kill
/reset
/spawn
/teleop_turtle/describe_parameters
/teleop_turtle/get_parameter_types
/teleop_turtle/get_parameters
/teleop_turtle/list_parameters
/teleop_turtle/set_parameters
/teleop_turtle/set_parameters_atomically
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
yahboom@yahboom-virtual-machine:~$
```

6.2, ros2 service call

Function: Call a specified service

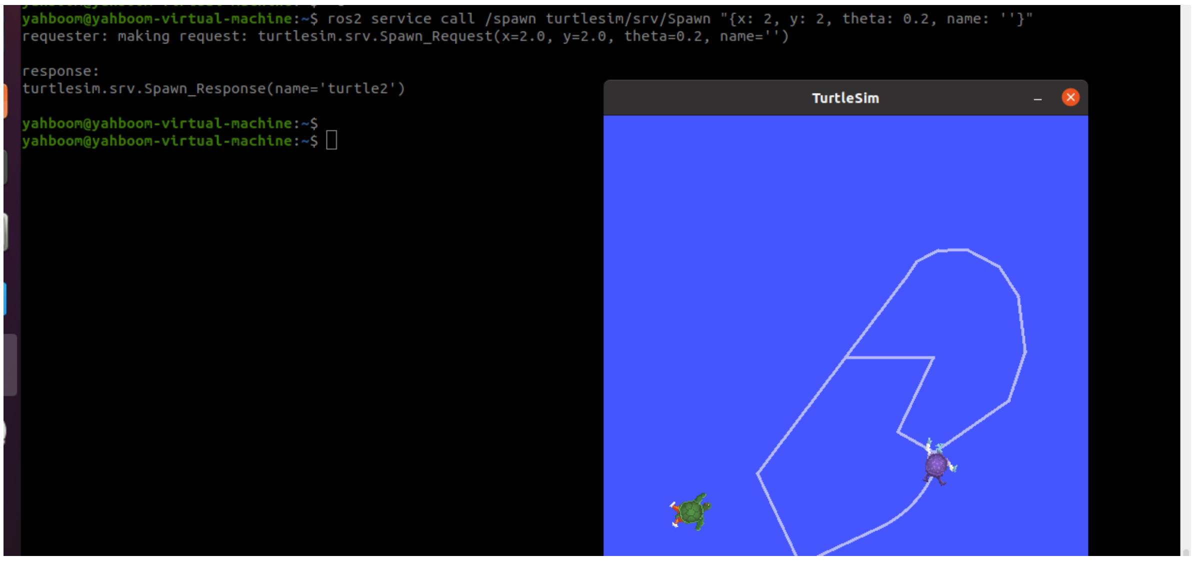
Format:

```
ros2 interface call service_name service_Type arguments
```

- service_name: The service to be called
- service_type: The service data type
- arguments: The parameters required to provide the service

For example, to call the turtle spawn service

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: 'turtle10'}"
```



```
yahboom@yahboom-virtual-machine:~$ ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: ''}"
requester: making request: turtlesim.srv.Spawn_Request(x=2.0, y=2.0, theta=0.2, name='')

response:
turtlesim.srv.Spawn_Response(name='turtle2')

yahboom@yahboom-virtual-machine:~$
yahboom@yahboom-virtual-machine:~$
```