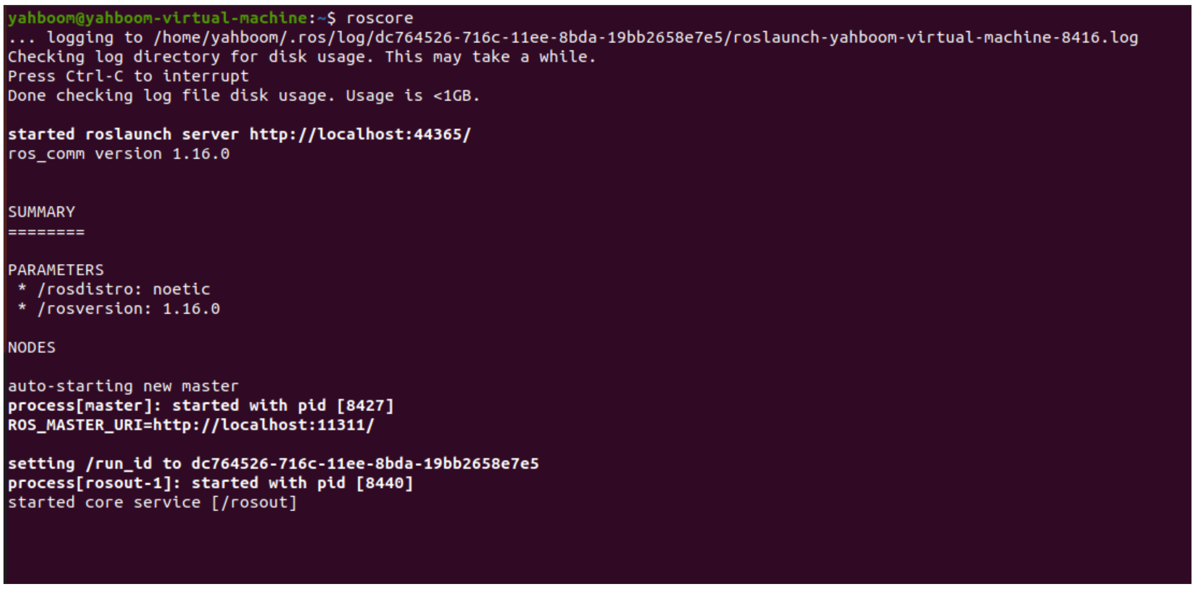
6.ROS node

In [1. Introduction to ROS]-[1.2.1. Computational graph level], we introduced the concept of nodes. In the previous section, the function package we created is the folder that stores the node program. You can use Python or C++ to write the node program, then compile it into an executable file, and finally run it as what we call the node program.

6.1 roscore

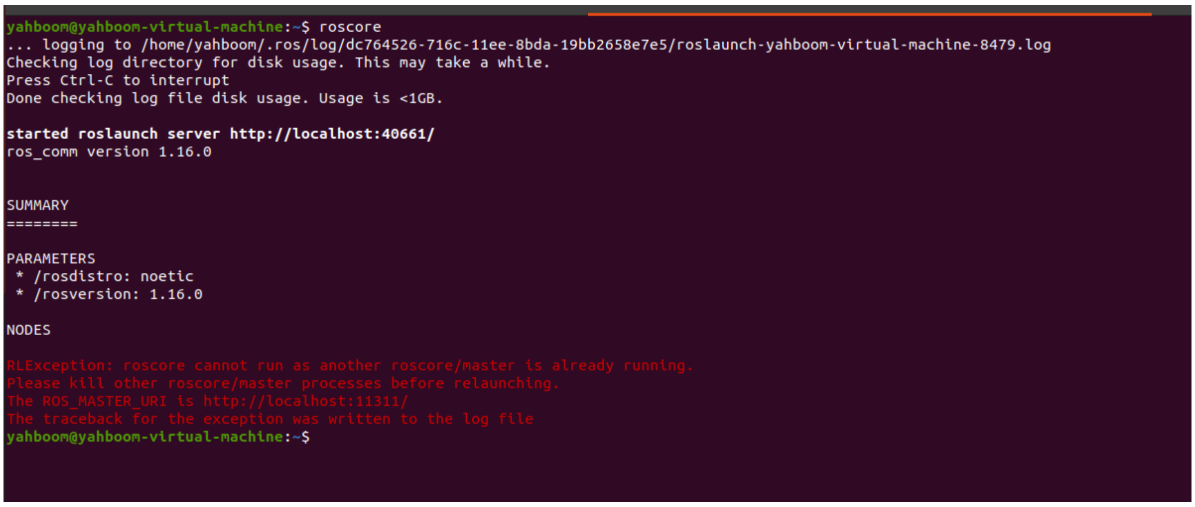
Before running all ros programs, you need to start roscore (this is not required when running the launch file later, roscore will be started when the launch file is started), enter in the terminal,

```
roscore
```



```
yahboom@yahboom-virtual-machine:~$ roscore
... logging to /home/yahboom/.ros/log/dc764526-716c-11ee-8bda-19bb2658e7e5/roslaunch-yahboom-virtual-machine-8416.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:44365/
ros_comm version 1.16.0


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [8427]
ROS_MASTER_URI=http://localhost:11311/

setting /run_id to dc764526-716c-11ee-8bda-19bb2658e7e5
process[rosout-1]: started with pid [8440]
started core service [/rosout]
```

Only one roscore can be run. If roscore is started in multiple terminals, it will prompt that roscore has been started, as shown in the figure below.

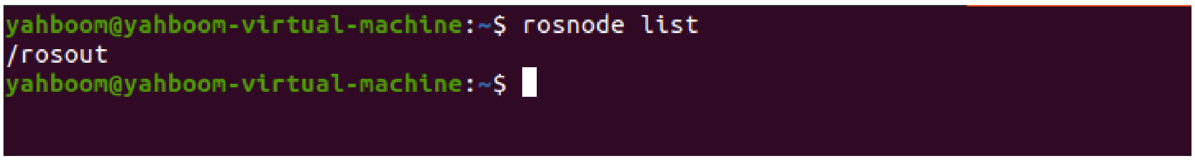


```
yahboom@yahboom-virtual-machine:~$ roscore
... logging to /home/yahboom/.ros/log/dc764526-716c-11ee-8bda-19bb2658e7e5/roslaunch-yahboom-virtual-machine-8479.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:40661/
ros_comm version 1.16.0


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.16.0

NODES

RLException: roscore cannot run as another roscore/master is already running.
Please kill other roscore/master processes before relaunching.
The ROS_MASTER_URI is http://localhost:11311/
The traceback for the exception was written to the log file
yahboom@yahboom-virtual-machine:~$
```

6.2 rosnode

After starting roscore, a node program is started. We have introduced several common tools of rosnode in the previous [3. ROS common command tools]-[3.1. Node rosnode].We can use **rosnode list** to view and query all currently running nodes, and enter in the terminal,

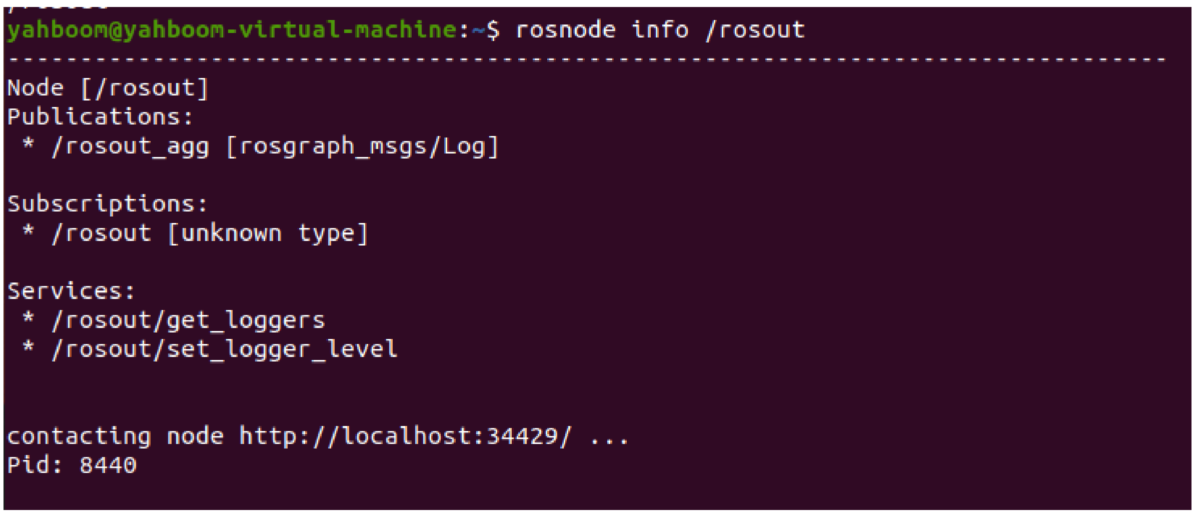
```
rosnode list
```



```
yahboom@yahboom-virtual-machine:~$ rosnode list
/rosout
yahboom@yahboom-virtual-machine:~$
```

Only one node is started here, which is /rosout. This is the node that we run after starting the roscore program. You can use rosnode info node_name to view the information of the node.(node_name represents the node name, modify it according to the actual node name that needs to be queried), enter the terminal,

```
rosnode info /rosout
```



```
yahboom@yahboom-virtual-machine:~$ rosnode info /rosout
--------------------------------------------------------------------------------
Node [/rosout]
Publications:
 * /rosout_agg [rosgraph_msgs/Log]

Subscriptions:
 * /rosout [unknown type]

Services:
 * /rosout/get_loggers
 * /rosout/set_logger_level


contacting node http://localhost:34429/ ...
Pid: 8440
```

As shown in the figure above, some relevant information about the node will be printed and listed, such as:

- Posted topics and related data types

  Publications:

  - /rosout_agg [rosgraph_msgs/Log]
- Subscribed topics and related data types

  Subscriptions:

  - /rosout [unknown type]
- Services provided and related data types

  Services:

  - /rosout/get_loggers
  - /rosout/set_logger_level

6.3 rosrun

Rosrun is the command to start the ros node program. The previous roscore is special. You can start it by inputting roscore in the terminal. However, most other ros node programs are started by rosrun. The command format is as follows:

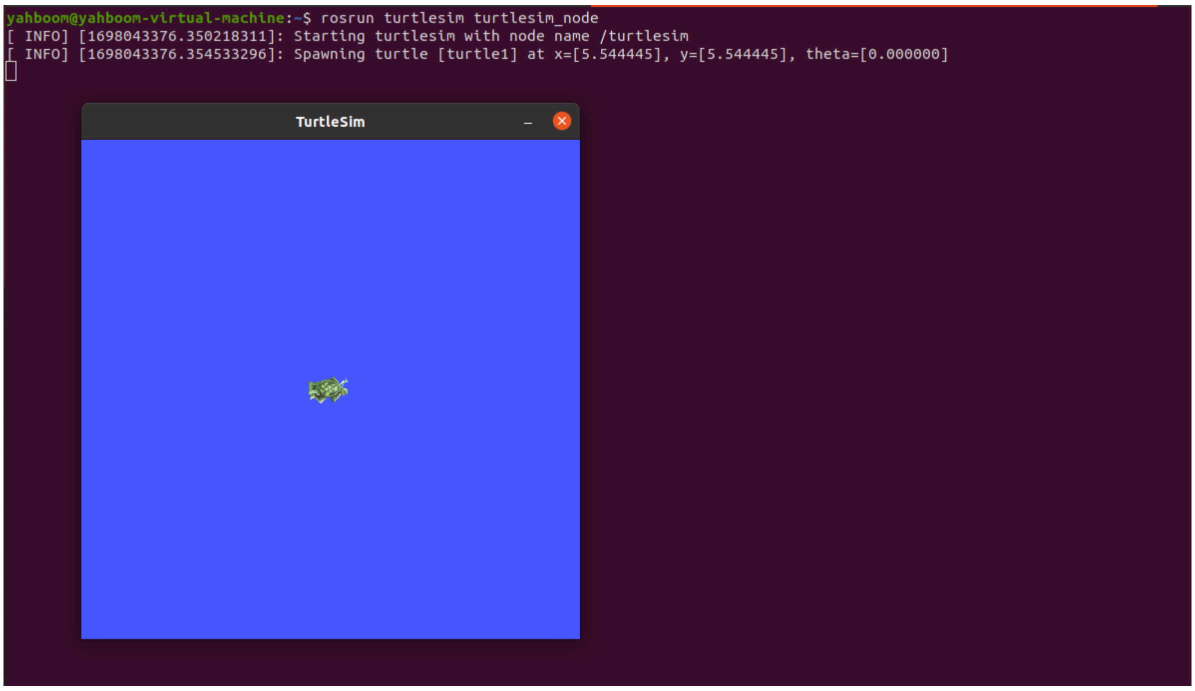
```
rosrun pkg_name executable_program
```

pkg_name: function package name, the name of the function package folder of the subsequent executable program

executable_program: The name of the executable program, which can be a file generated by C++ compilation, or a file written in Python with a .py executable file at the end.

For example, taking the classic little turtle as an example, after starting roscore, we enter in another terminal,

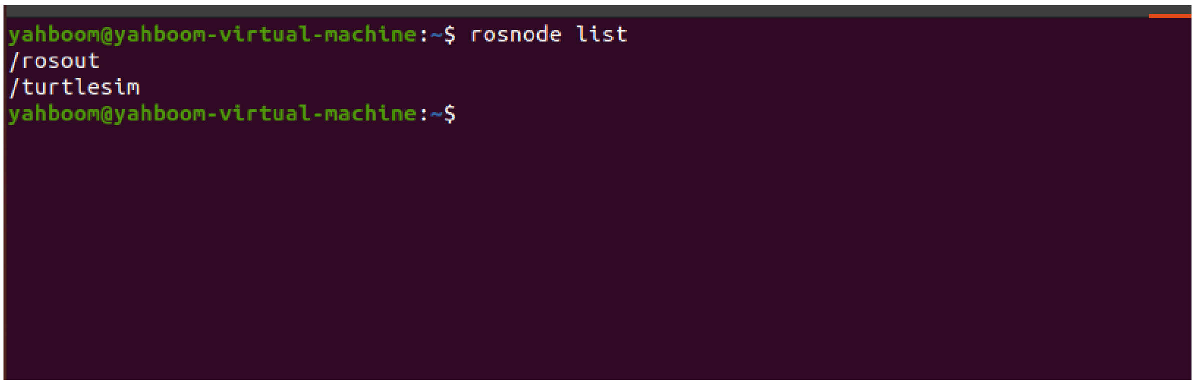
```
rosrun turtlesim turtlesim_node
```



```
yahboom@yahboom-virtual-machine:~$ rosrun turtlesim turtlesim_node
[ INFO] [1698043376.350218311]: Starting turtlesim with node name /turtlesim
[ INFO] [1698043376.354533296]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

After successful startup, a little turtle will appear. We can check the current nodes through rosnode list and enter in the terminal.

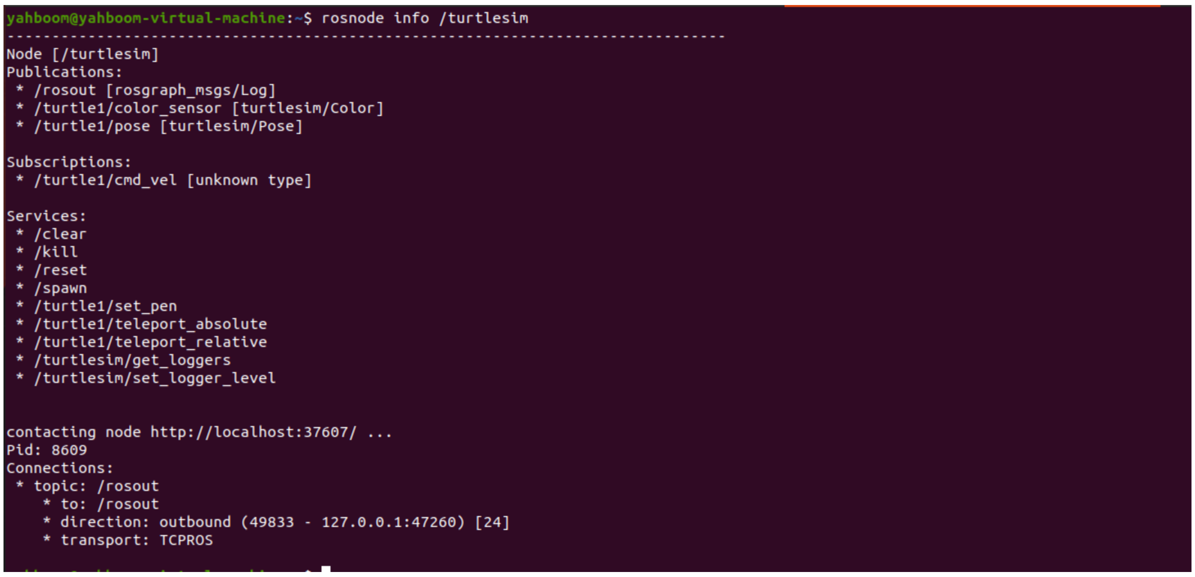
```
rosnode list
```



```
yahboom@yahboom-virtual-machine:~$ rosnode list
/rosout
/turtlesim
yahboom@yahboom-virtual-machine:~$
```

Compared with the previous /rosout, there is an additional /turtlesim. You can also use the rosnode info tool to view node information and enter it in the terminal.

```
rosnode info /turtlesim
```



```
yahboom@yahboom-virtual-machine:~$ rosnode info /turtlesim
--------------------------------------------------------------------------------
Node [/turtlesim]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/color_sensor [turtlesim/Color]
 * /turtle1/pose [turtlesim/Pose]

Subscriptions:
 * /turtle1/cmd_vel [unknown type]

Services:
 * /clear
 * /kill
 * /reset
 * /spawn
 * /turtle1/set_pen
 * /turtle1/teleport_absolute
 * /turtle1/teleport_relative
 * /turtlesim/get_loggers
 * /turtlesim/set_logger_level


contacting node http://localhost:37607/ ...
Pid: 8609
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound (49833 - 127.0.0.1:47260) [24]
    * transport: TCPROS
```

There will be more content here, but basically the content is similar. The [Publications] section explains which topics the node has published and the corresponding topic data types; [Subscriptions] section explains which topics the node subscribes to and the corresponding topic data types; the [Services] section explains what services the node provides.

Making more use of the rosnode tool to query running nodes and related information is a very important point in the process of debugging ros. For example, we wrote a program and ran it without problems and reported errors, but the topic communication between nodes did not run as we expected. At this time, you can use rosnode info to check whether it is caused by inconsistent topic names or other content. Only by running these tools flexibly can the problem be solved faster.