1.ROS Introduction

ROS wiki : http://wiki.ros.org/

ROS teaching: http://wiki.ros.org/ROS/Tutorials

ROS install: http://wiki.ros.org/melodic/Installation/Ubuntu

ROS (Robot Operating System) is an open source operating system suitable for robots. It provides the necessary services for the operating system, including hardware abstraction, underlying device control, implementation of commonly used functions, inter process messaging, and package management. It also provides the tools and library functions needed to obtain, compile, write, and run code across computers.

The main goal of ROS is to provide code reuse support for robot research and development. ROS is a distributed process (also known as "node") framework, which is encapsulated in program and feature packages that are easy to share and publish. ROS also supports a joint system similar to a code repository, which can also achieve engineering collaboration and publishing. This design can enable the development and implementation of a project to make completely independent decisions from the file system to the user interface (not limited by ROS). At the same time, all projects can be integrated with the basic tools of ROS.

1.1 Main characteristics of ROS

(1) Distributed architecture (where each work process is treated as a node and managed uniformly using a node manager),

(2) Multi language support (such as C++, Python, etc.),

(3) Good scalability (you can write a node or organize many nodes into a larger project through Rosslaunch)

(4) Open source (ROS follows the BSD protocol and is completely free for individual and commercial applications and modifications).

1.2 Overall Architecture of ROS

Open source community level: mainly including developer knowledge, code, and algorithm sharing.

File system level: used to describe the code and executable programs that can be found on the hard drive,

Computational graph level: Reflects the communication between processes, as well as between processes and systems.

1.2.1 Calculation diagram level

- node

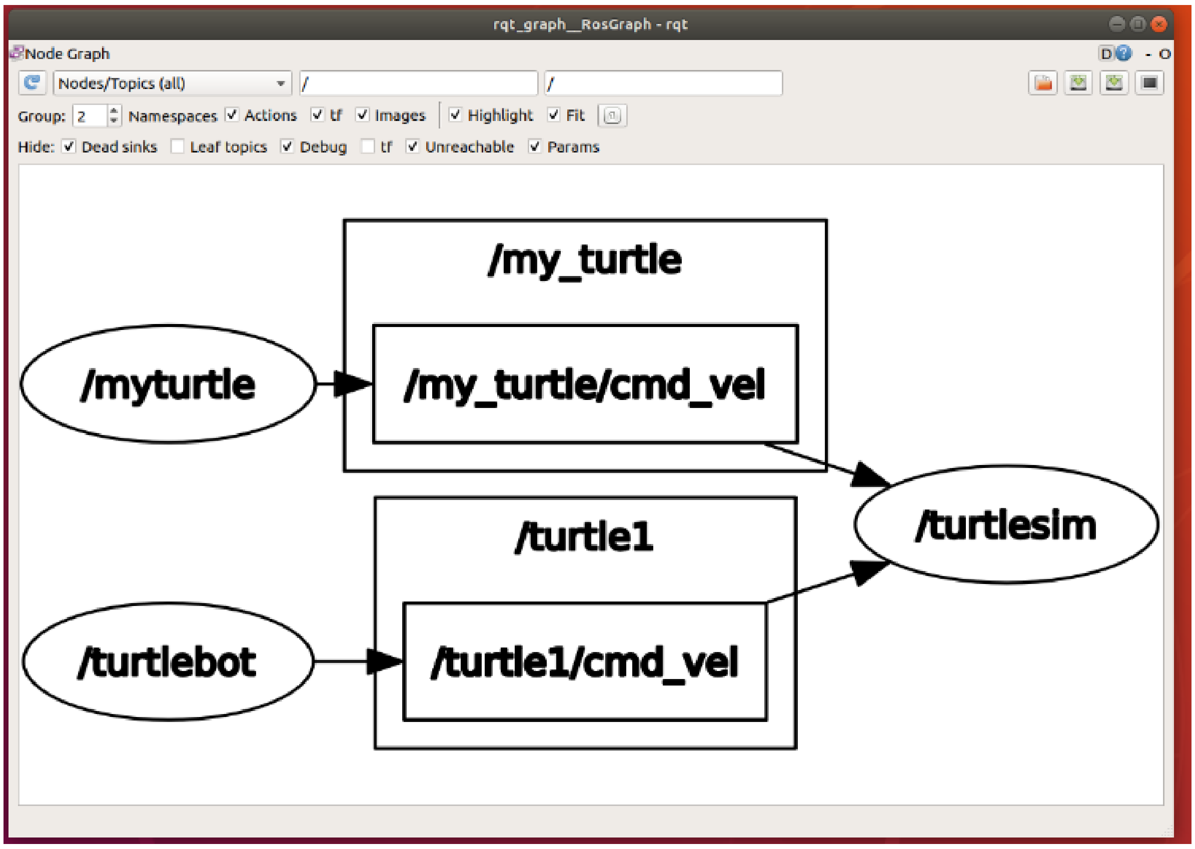
```
roscore
rosrun turtlesim turtlesim_node
```

The following command is the entire one. After entering a part, use the [Tab] key to complete it, and then modify the content

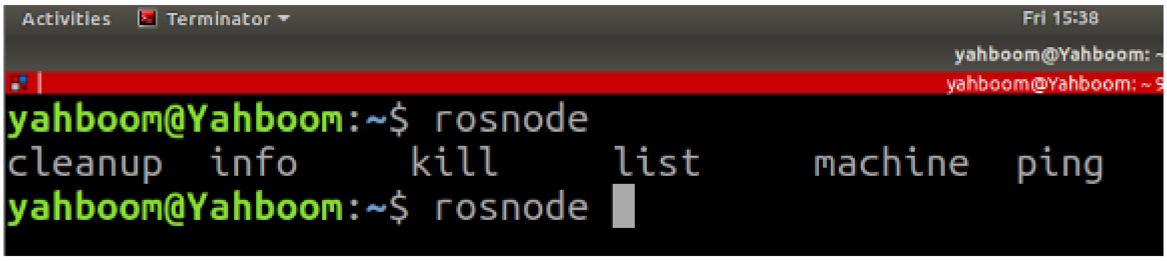
```
rosservice call /spawn "x: 3.0
y: 3.0
theta: 90.0
name: 'my_turtle'"
```

Nodes are the main computational execution processes. ROS is composed of many nodes.

```
rqt_graph
```



When we enter [rosnode] on the command line and double-click the Tab key, we will find that these words appear below the command line



```
yahboom@Yahboom:~$ rosnode
cleanup  info     kill     list     machine  ping
yahboom@Yahboom:~$ rosnode
```

ROS command-line tool rosnode:

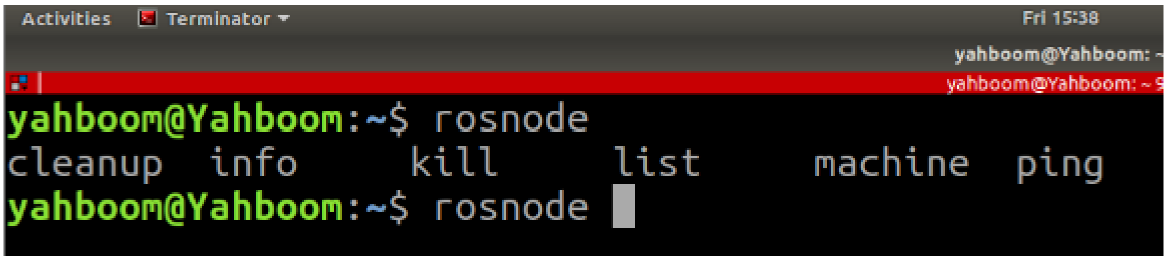
rosnode command	role
rosnode list	Query all currently running nodes
rosnode info node_name	Display detailed information of this node
rosnode kill node_name	End a node

rosnode command	role
rosnode ping	Test whether the node is alive
rosnode machine	List nodes running on specific machines or listed machines
rosnode cleanup	Clear the registration information of non runnable nodes

When developing and debugging, it is often necessary to view the current node and node information, so please remember these commonly used commands. If you can't remember, you can also check the usage of the rosnode command through the rosnode help.

- message

Nodes achieve logical connections and data exchange between each other through messages.When we enter [rosmsg] on the command line and double-click the Tab key, we will find that these words appear below the command line



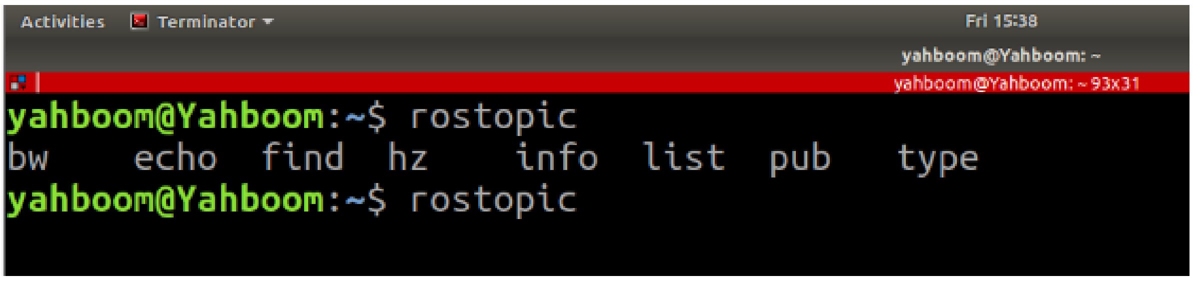
```
yahboom@Yahboom:~$ rosnode
cleanup  info     kill     list     machine  ping
yahboom@Yahboom:~$ rosnode
```

ROS command-line tool rosmsg:

rosmsg command	role
rosmsg show	Display a message field
rosmsg list	List all messages
rosmsg package	List all feature pack messages
rosmsg packages	List all feature packages with this message
rosmsg md5	Display MD5 verification value for a message

- Topic

A topic is a way of delivering messages (publish/subscribe). Every message should be posted to the corresponding topic, and each topic is strongly typed.The topic messages of ROS can be transmitted using TCP/IP or UDP, and the default transmission method used by ROS is TCP/IP. TCP based transmission becomes TCPROS, which is a long connection method; UDP based transmission, known as UDPROS, is a low latency and efficient transmission method, but it is prone to data loss and is suitable for remote operations.When we enter 'rostopic' on the command line and double-click the Tab key, we will find that these words appear below the command line



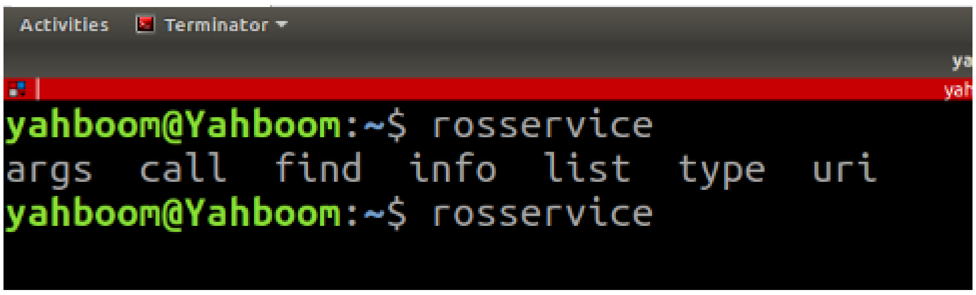
```
yahboom@Yahboom:~$ rostopic
bw     echo   find   hz     info   list   pub    type
yahboom@Yahboom:~$ rostopic
```

ROS command-line tool rostopic

rostopic command	role
rostopic bw /topic	Display the bandwidth used by the theme
rostopic echo /topic	Output the message corresponding to the topic to the screen
rostopic find message_type	Find topics by type
rostopic hz /topic	Display the frequency of topic publishing
rostopic info /topic	Output information about the topic
rostopic list	Output activity topic list
rostopic pub /topic type args	Publish data to a topic
rostopic type /topic	Output Theme Type

- give service to

The service used for the request response model must also have a unique name. When a node provides a service, all nodes can communicate with it by using code written by ROS clients.When we enter 'rossservice' on the command line and double-click the Tab key, we will find that these words appear below the command line



```
yahboom@Yahboom:~$ rosservice
args  call  find  info  list  type  uri
yahboom@Yahboom:~$ rosservice
```

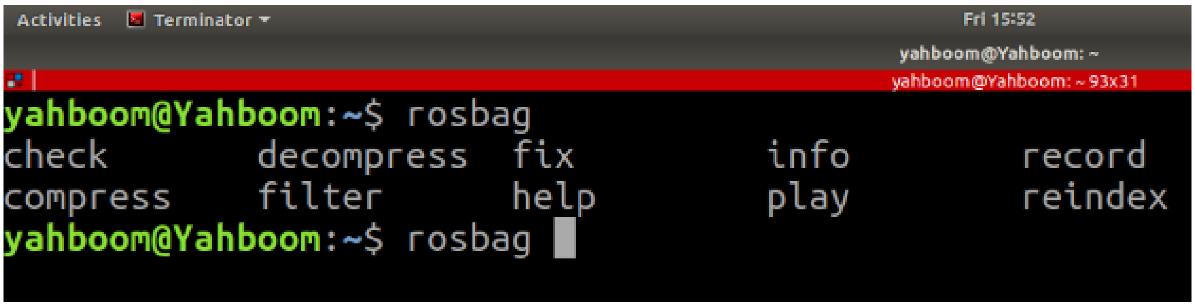
ROS command-line tool roseservice:

rosservice command	role
rosservice args /service	effectRosservice args/service displays service parameters
rosservice call /service	Requesting service with input parameters
rosservice find /service	Find topics by type
rosservice info /service	Display information for the specified service

rosservice command	role
rosservice list	Display active service information
rosservice uri /service	Display ROSRPC URI Service
rosservice type /service	Display service type

- Message logging package

Message recording package is a file format used to save and replay ROS message data, stored in a. bag file. It is an important mechanism for storing data.When we enter [rostag] on the command line and double-click the Tab key, we will find that these words appear below the command line



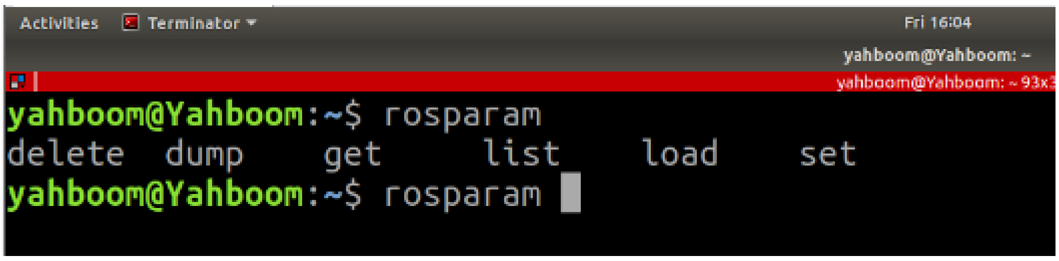
```
yahboom@Yahboom:~$ rosbag
check       decompress  fix         info        record
compress    filter      help        play        reindex
yahboom@Yahboom:~$ rosbag 
```

ROS command-line tool rosbag:

rosbag command	role
check	Determine whether a package can be processed in the current system or migrated.
decompress	Compress one or more package files.
filter	Unzip one or more package files.
fix	Fix the message in the package file for playback in the current system.
help	Obtain relevant command guidance and help information.
info	Summarize the content of one or more package files.
play	Playback the content of one or more package files in a time synchronized manner.
record	Record a package file with the content of the specified topic.
reindex	Reindex one or more package files.

- parameter server

The parameter server is a shared multivariate dictionary that can be accessed through the network and stored on the node manager through keywords.When we enter 'rosparam' on the command line and double-click the Tab key, we will find that these words appear below the command line



```
yahboom@Yahboom:~$ rosparam
delete  dump    get     list    load    set
yahboom@Yahboom:~$ rosparam
```

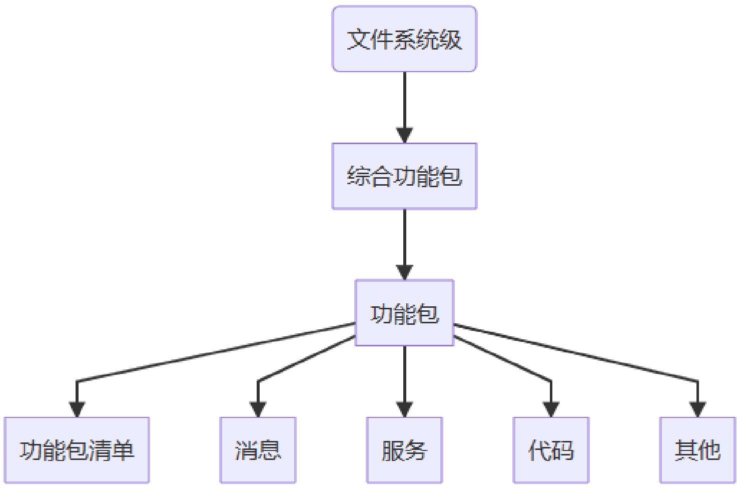
ROS command-line tool rosparam:

rosparam command	role
rosparam delete parameter	Remove Parameter
rosparam dump file	Write parameters from the parameter server to a file
rosparam get parameter	Obtain parameter values
rosparam list	List parameters in the parameter server
rosparam load file	Load parameters from file to parameter server
rosparam set parameter value	Set parameters

- Node Manager (Master)

The node manager is used for registering and searching topics and service names. If there is no node manager in the entire ROS system, there will be no communication between nodes.

1.2.2 File System Level



Dependency relationships can be configured between feature packages. If Function Package A relies on Function Package B, then when building the system in ROS, B must be built earlier than A, and A can use the header and library files in B.The concept of file system level is as follows:

- List of feature packages:

This list indicates the dependencies of the feature pack, source file compilation flag information, and so on. The package.xml file in the feature pack is a list of feature packs.

- Feature Pack:

Function packages are the basic form of software organization in ROS systems, including running nodes and configuration files.

ROS package related commands

rospack command	role
rospack help	Display the usage of rospack
rospack list	列出本机所有packageList all packages on this machine
rospack depends [package]	Display the dependent packages of the package
rospack find [package]	Locate a package
rospack profile	Refresh location records for all packages

- Comprehensive function package

Organize several functional packages together to form a comprehensive functional package.

- message type

When sending messages between nodes in ROS, message explanations need to be provided in advance. ROS provides standard type messages, which can also be defined by oneself. The description of the message type is stored in the msg file under the feature pack.

- Service Type

Defined the data structure provided by each process in the ROS system regarding service requests and responses.1.2.3. Open source community level

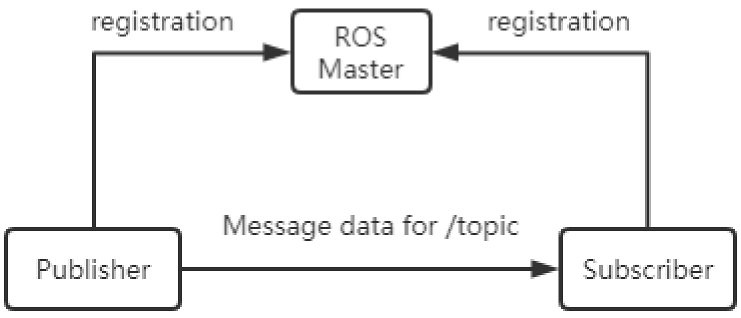
1.2.3. Open source community level

- Distribution: The ROS distribution is a series of comprehensive feature packs that can be installed independently and have version numbers. The ROS distribution plays a similar role as the Linux distribution. This makes ROS software installation easier and enables consistent versions to be maintained through a collection of software.
- Software library: ROS relies on websites or host services that share open source code and software library, where different institutions can publish and share their own robot software and programs.
- ROS Wiki: ROS Wiki is the main forum used to record information about ROS systems. Anyone can register an account, contribute their own files, provide corrections or updates, write tutorials, and engage in other activities.
- Bug Ticket System: If you discover problems or want to propose a new feature, ROS provides this resource to do so.
- Mailing list: ROS user mailing list is the main communication channel about ROS, which can communicate various questions or information from ROS software updates to ROS software usage like a forum
- OS Q&A (ROS Answer): Users can use this resource to ask questions

1.3 Communication Mechanism

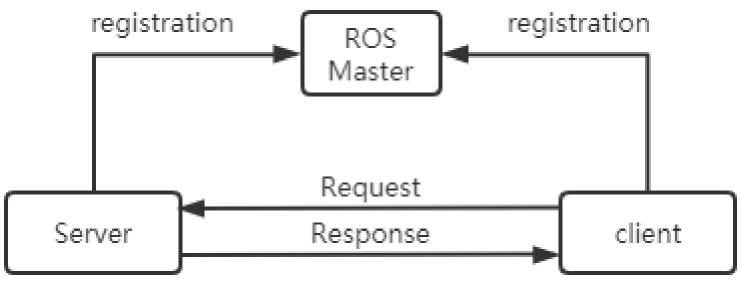
1.3.1、Topic

The asynchronous publish subscribe communication mode is widely used in ROS. Topic is generally used for one-way, message flow communication. Topic generally has a strong type definition: a topic of one type can only accept/send messages of a specific data type (message type). Publisher is not required for type consistency, but upon acceptance, the subscriber will check the MD5 of the type and report an error.



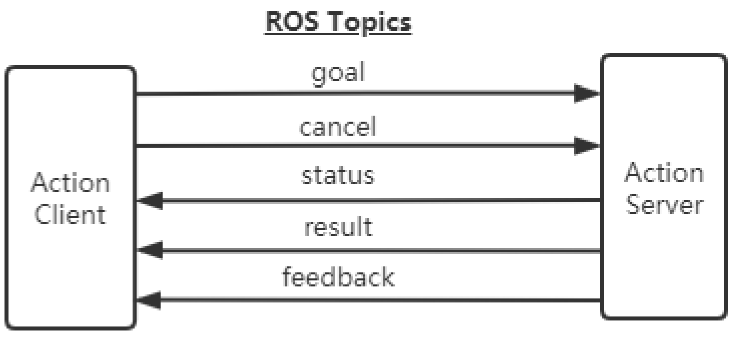
1.3.2、Service

Service is used to handle synchronous communication in ROS communication, using server/client semantics. Each service type has two parts: request and response. For servers in the service, ROS does not check for name conflicts. Only the last registered server will take effect and establish a connection with the client.



1.3.3、Action

Action is composed of multiple topics used to define tasks, including goals, feedback on task execution process status, and results. Compiling an action will automatically generate 7 structures: Action, ActionGoal, ActionFeedback, ActionResult, Goal, Feedback, and Result structures.



Characteristics of Action:

- A Question and Answer Communication Mechanism
- With continuous feedback

- Can be terminated during the task
- Implementation of message mechanism based on ROS

Interface for Action:

- goal：Publish Task Objectives
- cancel：Request to cancel task
- status：Notify client of current status
- feedback：Monitoring data for periodic feedback task operation
- result：Send the execution results of the task to the client and publish them only once.

Comparison of communication mode characteristics

characteristic	**Topic**	**Service**	**Action**
response mechanism	no	Result response	Progress response
synchronism	synchronous	synchronous	synchronous
communication model	`Publisher`，`Subscriber`	`Client`，`Server`	`Client`，`Server`
Node correspondence	Many to many	Client to Server	Client to Server

1.4 Common Components

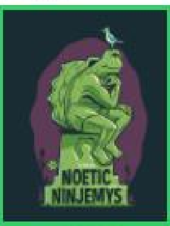
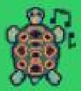
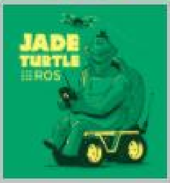
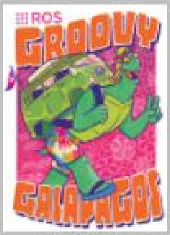
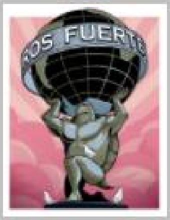
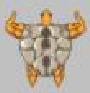
Launch startup file; TF coordinate transformation; Rviz； Gazebo； QT toolbox; Navigation; MoveIt!Launch: Launch File is a way in ROS to start multiple nodes simultaneously. It can also automatically start the ROS Master node manager and implement various configurations for each node, providing great convenience for the operation of multiple nodes.TF coordinate transformation: There are often a large number of component elements in the robot body and the working environment of the robot, which involve the position and posture of different components in robot design and application. TF is a functional package that allows users to track multiple coordinate systems over time. It uses a tree data structure to buffer and maintain coordinate transformation relationships between multiple coordinate systems based on time, which can help developers at any time Complete the transformation of points, vectors, and other coordinates between coordinate systems.QT Toolbox: In order to facilitate visual debugging and display, ROS provides a Qt architecture backend graphical tool kit - rqt_ common_ Plugins, which includes many practical tools: log output tool (rqt_console), computational graph visualization tool (rqt_graph), data plotting tool (rqt_plot), parameter dynamic configuration tool (rqt-reconfigure)

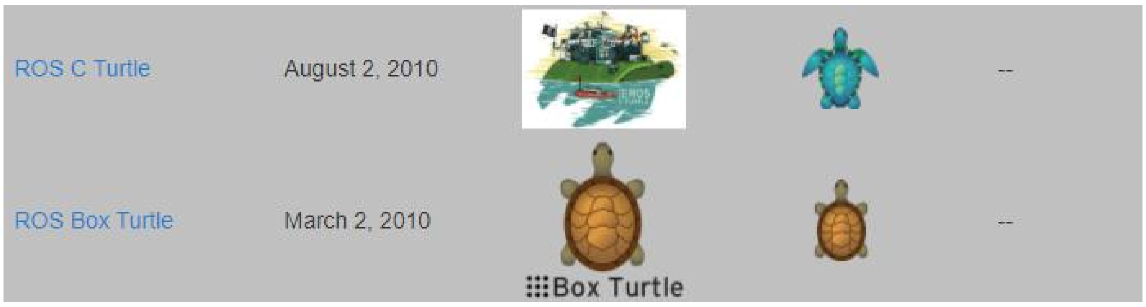
Rviz: rviz is a 3D visualization tool, which is well compatible with various robot platforms based on ROS software framework. In rviz, XML can be used to describe the dimensions, masses, positions, materials, joints, and other attributes of robots, surrounding objects, and any other physical objects, and presented in the interface. At the same time, rviz can also display real-time information about robot sensors, robot motion status, and changes in the surrounding environment through graphical means.Gazebo: Gazebo is a powerful 3D physics simulation platform with powerful physics engines, high-quality graphics rendering, convenient programming and graphics interfaces, and most importantly, it has open source and free features. Although the robot model in Gazebo is the same as the model used in rviz, it is necessary to incorporate the physical properties of the robot and its surrounding environment, such as mass, friction coefficient, elasticity coefficient, etc., into the model. The sensor information of the robot can also be added to the simulation environment through plugins and displayed in a visual manner.

Navigation: navigation is the two-dimensional navigation function package of ROS. In brief, it is based on the information flow of odometer and other sensors input and the global position of the robot, and through the navigation algorithm, it can calculate safe and reliable robot speed control commands.Moveit：Moveit！ The feature pack is the most commonly used tool kit, mainly used for trajectory planning. Moveit！ The configuration assistant is crucial for configuring files that need to be used in planning.

1.5 Release version

Reference link：http://wiki.ros.org/Distributions

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Noetic Ninjemys (**Recommended**)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013			May, 2015
ROS Groovy Galapagos	December 31, 2012			July, 2014
ROS Fuerte Turtle	April 23, 2012			--
ROS Electric Emys	August 30, 2011			--
ROS Diamondback	March 2, 2011			--



ROS C Turtle	August 2, 2010			--
ROS Box Turtle	March 2, 2010			--

The ROS distribution refers to the version of the ROS software package, which is similar in concept to Linux distributions such as Ubuntu. The purpose of launching the ROS distribution version is to enable developers to use a relatively stable code library until they are ready to upgrade all content. Therefore, after the release of each release, ROS developers usually only fix the bugs in that version and provide a small number of improvements for the core software package. As of October 2019, the version names, release times, and version lifecycles of the main release versions of ROS are shown in the table below

Version Name	**Release Date**	**Version lifecycle**	**Operating System Platform**
ROS Noetic NinjemysROS Noetic Ninjemys	2020年5月2020年5月	2025年5月2025年5月	Ubuntu 20.04Ubuntu 20.04
ROS Melodic MoreniaROS Melodic Morenia	2018年5月23日2018年5月23日	2023年5月2023年5月	Ubuntu 17.10, Ubuntu 18.04, Debian 9, Windows 10Ubuntu 17.10, Ubuntu 18.04, Debian 9, Windows 10
ROS Lunar LoggerheadROS Lunar Loggerhead	2017年5月23日2017年5月23日	2019年5月2019年5月	Ubuntu 16.04, Ubuntu 16.10, Ubuntu 17.04,Debian 9Ubuntu 16.04, Ubuntu 16.10, Ubuntu 17.04,Debian 9
ROS Kinetic KameROS Kinetic Kame	2016年5月23日2016年5月23日	2021年4月2021年4月	Ubuntu 15.10, Ubuntu 16.04, Debian 8Ubuntu 15.10, Ubuntu 16.04, Debian 8
ROS Jade TurtleROS Jade Turtle	2015年5月23日2015年5月23日	2017年5月2017年5月	Ubuntu 14.04, Ubuntu 14.10, Ubuntu 15.04Ubuntu 14.04, Ubuntu 14.10, Ubuntu 15.04
ROS Indigo IglooROS Indigo Igloo	2014年7月22日2014年7月22日	2019年4月2019年4月	Ubuntu 13.04, Ubuntu 14.04Ubuntu 13.04, Ubuntu 14.04
ROS Hydro MedusaROS Hydro Medusa	2013年9月4日2013年9月4日	2015年5月2015年5月	Ubuntu 12.04, Ubuntu 12.10, Ubuntu 13.04Ubuntu 12.04, Ubuntu 12.10, Ubuntu 13.04
ROS Groovy GalapagosROS Groovy Galapagos	2012年12月31日2012年12月31日	2014年7月2014年7月	Ubuntu 11.10, Ubuntu 12.04, Ubuntu 12.10Ubuntu 11.10, Ubuntu 12.04, Ubuntu 12.10

Version Name	Release Date	Version lifecycle	Operating System Platform
ROS Fuerte TurtleROS Fuerte Turtle	2012年4月23日2012年4月23日	----	Ubuntu 10.04, Ubuntu 11.10, Ubuntu 12.04Ubuntu 10.04, Ubuntu 11.10, Ubuntu 12.04
ROS Electric EmysROS Electric Emys	2011年8月30日2011年8月30日	----	Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04, Ubuntu 11.10Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04, Ubuntu 11.10
ROS DiamondbackROS Diamondback	2011年3月2日2011年3月2日	----	Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04
ROS C TurtleROS C Turtle	2010年8月2日2010年8月2日	----	Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04, Ubuntu 10.10Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04, Ubuntu 10.10
ROS Box TurtleROS Box Turtle	2010年3月2日2010年3月2日	----	Ubuntu 8.04, Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04Ubuntu 8.04, Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04