7.1 Introduction of ROS

Note: In the image operating system provided by Yahboom, the APP remote control process is enabled by default, in order to avoid multiple occupations of internal resources, causing some functions to fail to operate normally.

Before you running the code of this course, please follow the following to close the APP remote control process.

If you want to permanently close the function of the APP that starts automatically after booting, execute the following command:

```
sudo systemctl disable jetbotmini_start
```

If you want to permanently open the function of the APP that starts automatically after booting, execute the following command:

```
sudo systemctl enable jetbotmini_start
```

If you do not restart Jetbotmini to restart the APP function, execute the following command:

```
sudo systemctl stop jetbotmini_start
sudo systemctl start jetbotmini_start
```

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

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

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

ROS（Robot Operating System）It is an open source operating system suitable for robots. It provides some necessary services to the operating system, including hardware abstraction, low-level device control, implementation of common functions,*message-passing*, and package management. It also provides tools and library functions for obtaining, compiling, writing, and running code across computers.

The goal of ROS is to provide code support for robot research and development. ROS is a distributed process framework (also known as "nodes"). These processes are encapsulated in packages and function packages that are easy to share and release.

ROS also supports a joint system similar to a code repository, which can realize project collaboration and release.

7.1.1 ROS Feature

（1）Distributed architecture (each work process is regarded as a node, and the node manager is used for unified management),

（2）Support multiple programming languages（such as C++, Python, etc.),

（3）Good scalability (one node can be written, or many nodes can be organized into a larger project through roslaunch),

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

7.1.2 ROS overall architecture

The Community level: mainly includes developer knowledge, code, and algorithm sharing.

The Filesystem level: be used to describe the codes and executable programs that can be found on the hard disk,

The Computational Graph level: reflects the communication between process and process, process and system.

1. The Computational Graph level

- nodes

```
roscore
rosrun turtlesim turtlesim_node
```

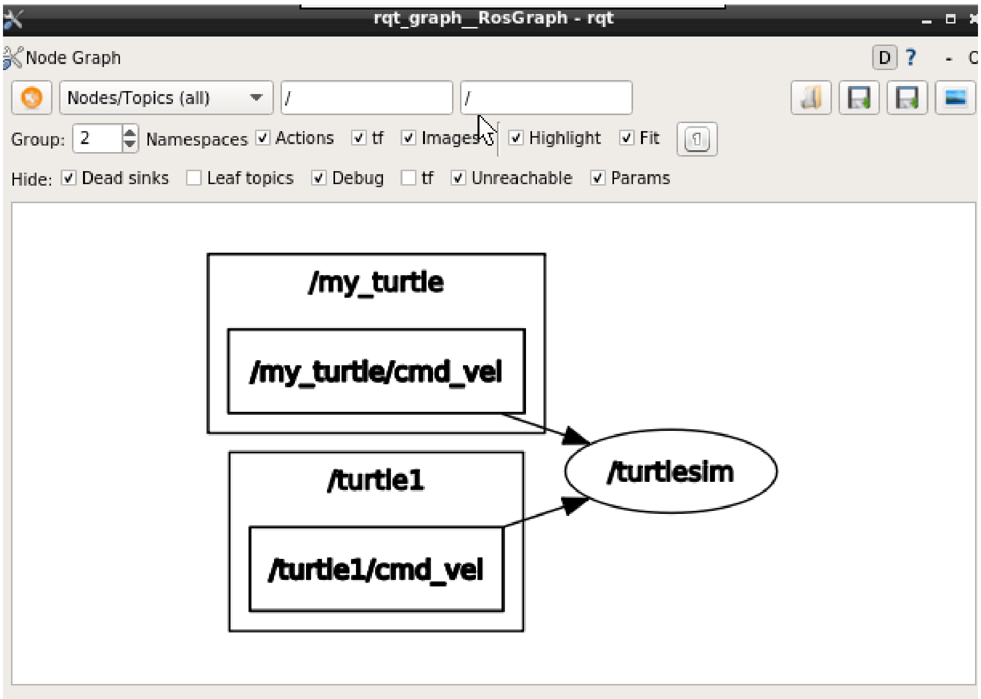
The following command is the entire one.

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'"
```

The node is the main calculation execution process. ROS is composed of many nodes.

```
rqt_graph
```



rqt_graph__RosGraph - rqt

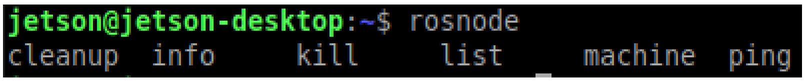
Node Graph

Nodes/Topics (all) / /

Group: 2 Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☐ Leaf topics ☑ Debug ☐ tf ☑ Unreachable ☑ Params

When we input 【rosnode】 in the command line, and then double-click the `Tab` key, we will find that several words appear below the command line.



```
jetson@jetson-desktop:~$ rosnode
cleanup  info      kill      list      machine  ping
```

ROS command line tool rosnode:

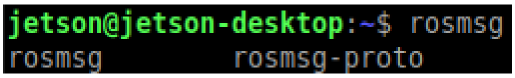
rosnode command	function
rosnode list	Query all nodes currently running
rosnode info node_name	Display the detailed information of the node
rosnode kill node_name	End a node
rosnode ping	Test whether the node is alive
rosnode machine	List the nodes running on a specific machine or list machine
rosnode cleanup	Clear the registration information of inoperable nodes

When developing and debugging, you often need to view the current node and node information, so please remember these common commands. If you can't remember, you can also check the usage of the `rosnode` command through `rosnode help`.

- message

  The nodes realize logical connection and data exchange with each other through messages.

  When we input【rosmsg】 in the command line, and then double-click the `Tab` key, we will find that several words appear below the command line.



```
jetson@jetson-desktop:~$ rosmsg
rosmsg          rosmsg-proto
```

ROS command line tool rosmsg:

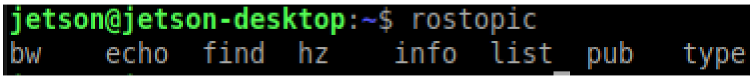
rosmsg command	function
rosmsg show	Show a message field
rosmsg list	List all messages
rosmsg package	List all feature pack messages
rosmsg packages	List all the feature packages with the message
rosmsg md5	Display the MD5 check value of a message

- Topic

The topic is a way of delivering messages (publish/subscribe). Each message must be posted on the corresponding topic.

ROS topic messages can be transmitted using TCP/IP or UDP, and the default transmission method used by ROS is TCP/IP. Based TCP transmission becomes TCPROS, which is a long connection method; based UDP transmission becomes UDPROS, which is a low-latency and high-efficiency transmission method, but it is easy to lose data and is suitable for remote operation.

When we input 【rostopic】 in the command line, and then double-click the `Tab` key, we will find that several words appear below the command line.



```
jetson@jetson-desktop:~$ rostopic
bw     echo   find   hz     info   list   pub    type
```

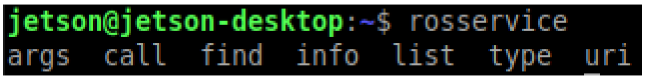
ROS command line tool rostopic：

rostopic command	function
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 topic publishing frequency
rostopic info /topic	Output information about the topic
rostopic list	Output a list of active topics
rostopic pub /topic type args	Publish data to topics
rostopic type /topic	Output theme type

- service

The service is used in the request response model and it must have a unique name. When a node provides a certain service, all nodes can communicate with it by using the code written by the ROS client.

When we input【rostopic】 in the command line, and then double-click the `Tab` key, we will find that several words appear below the command line.



```
jetson@jetson-desktop:~$ rosservice
args   call   find   info   list   type   uri
```

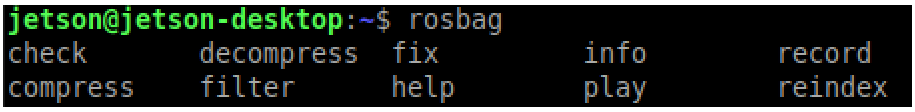
ROS command line tool rosservice：

rosservice command	function
rosservice args /service	Display service parameters
rosservice call /service	Request service with input parameters
rosservice find /service	Find topics by type
rosservice info /service	Display information about the specified service
rosservice list	Display active service information
rosservice uri /service	Show ROSRPC URI service
rosservice type /service	Display service type

- Message recording package

The message recording package is a file format used to save and play back ROS message data, and it is saved in a .bag file. It is an important mechanism for storing data.

When we input【rosbag】 in the command line, and then double-click the `Tab` key, we will find that several words appear below the command line.



```
jetson@jetson-desktop:~$ rosbag
check       decompress  fix         info        record
compress    filter      help        play        reindex
```

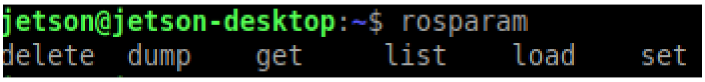
ROS command line tool rosbag:

rosbag command	function
check	Determine whether a package can be carried out in the current system, or whether it can be migrated.
decompress	Compress one or more package files.
filter	Extract one or more package files.
fix	Fix the message in the package file so that it can be played in the current system.
help	Get help information about related commands.
info	Summarize the contents of one or more package files.
play	Play back the contents of one or more package files in a time-synchronized manner.
record	Record a package file with the content of the specified topic.
reindex	Re-index one or more package files.

- Parameter server

The parameter server is a shared multi-variable dictionary that can be accessed via the network, and is stored on the node manager by keywords.

When we input【rosparam】 in the command line, and then double-click the `Tab` key, we will find that several words appear below the command line.



```
jetson@jetson-desktop:~$ rosparam
delete  dump    get     list    load    set
```

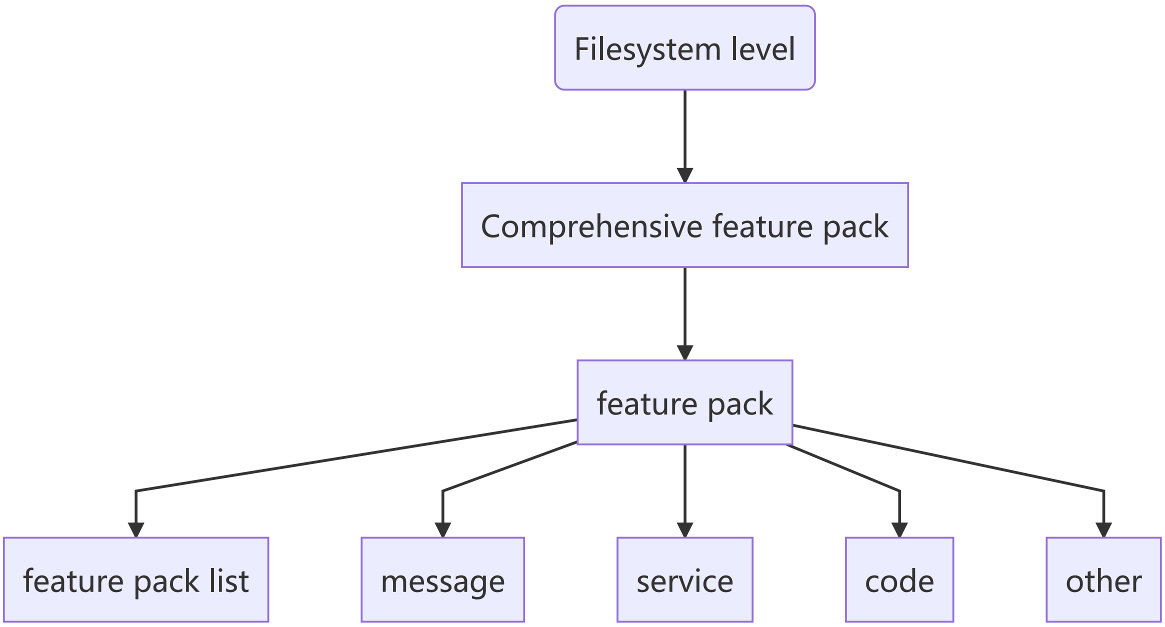
ROS command line tool rosparam:

rosparam command	function
rosparam delete parameter	Delete parameter
rosparam dump file	Write the parameters in the parameter server to the file
rosparam get parameter	Get parameter value
rosparam list	List the parameters in the parameter server
rosparam load file	Load parameters from file to parameter server
rosparam set parameter value	Setting parameters

- Node Manager （Master）

The node manager is used for the registration and search of themes and service names. If there is no node manager in the entire ROS system, there will be no communication between nodes.

2. Filesystem level



Dependencies can be configured between function packages. If the function package A depends on the function package B, then when the ROS build system, B must be built earlier than A, and A can use the header files and library files in B.

The concept of the file system level is as follows:

- Feature package list：

This list is to indicate the dependency of the function package, source file compilation flag information, etc. The package.xml file in the feature package is a feature package list.

- Feature package:

The function package is the basic form of software organization in the ROS system, including running nodes and configuration files.

ROS package related commands

rospack command	function
rospack help	Show usage of rospack
rospack list	List all packages of this machine
rospack depends [package]	Show package dependencies
rospack find [package]	Locate a package
rospack profile	Refresh location records of all packages

- Comprehensive function package

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

- Message type

When sending messages between nodes in ROS, message descriptions are required in advance. Standard types of messages are provided in ROS, and they can also be defined by themselves. The description of the message type is stored in the msg file under the function package.

- Service type

Defines the data structure of service requests and responses provided by each process in the ROS system.

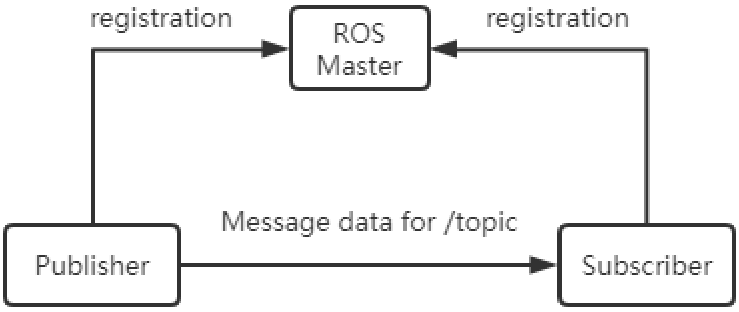
3. **The Community level**

- The ROS release is a series of comprehensive function packages with version numbers that can be installed independently. ROS distributions play a similar role as Linux distributions. This makes the installation of ROS software easier, and can maintain a consistent version through a software collection.
- Repository: ROS relies on websites or hosting services that share open source code and software libraries, where different organizations can publish and share their robot software and programs.
- ROS Wiki: ROS Wiki is the main forum for recording information about the ROS system. Anyone can register for an account, contribute their own files, provide corrections or updates, write tutorials, and other actions.
- Bug Ticket System: If you find a problem or want to propose a new idea, ROS can provide resources for you to implement these features.
- Mailing list: The ROS user mailing list is the main communication channel for ROS.
- ROS Answer: Users can use this resource to ask questions

7.1.3 Communication mechanism

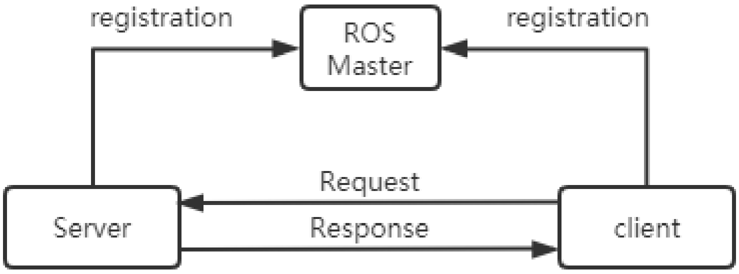
1. **Topic**

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 type of topic can only accept/send messages of a specific data type (message type). Publisher is not required for type consistency, but when accepting, the subscriber will check the md5 of the type and report an error.



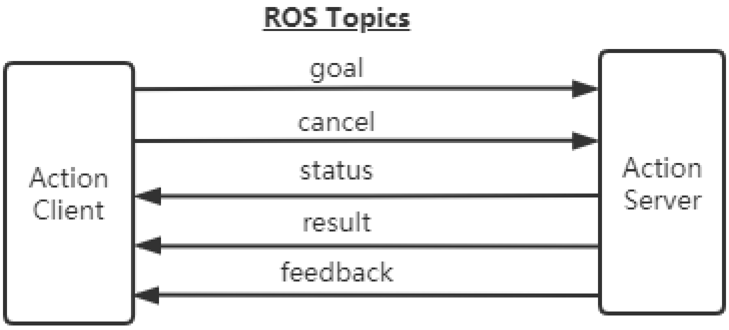
2. **Service**

service is used to process synchronous communication in ros communication, using server/client semantics. Each service type has two parts: request and response. For the server in the service, ros will not check for the same name (name conflict). Only the last registered server will take effect and establish a connection with the client.



3. Action

Actions are composed of multiple topics to define tasks. Task definitions include goals (Goal), task execution process status feedback (Feedback) and results (Result), etc. Compiling action will automatically generate 7 structures: Action, ActionGoal, ActionFeedback, ActionResult, Goal, Feedback, Result structure.



Action features

- A question and answer communication mechanism
- With continuous feedback
- Can be terminated during the mission
- Implementation based ROS message mechanism

Action interface

- goal: publish the task goal
- cancel: request to cancel the task
- status: notify the client of the current status
- feedback: feedback task running monitoring data
- result: send the execution result of the task to the client, only published once.

Comparison of characteristics of communication modes

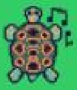
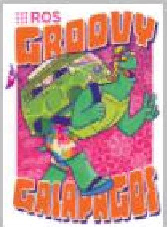
Features	Topic	Service	Action
Response mechanism	no	Result response	Progress response Result response
Synchronization	Asynchronous	Synchronize	Asynchronous
Communication model	`Publisher`, `Subscriber`	`Client`, `Server`	`Client`, `Server`
Node correspondence	Multiple to Multiple y	Multiple (Client) to one (Server)	Multiple (Client) to oneServer)

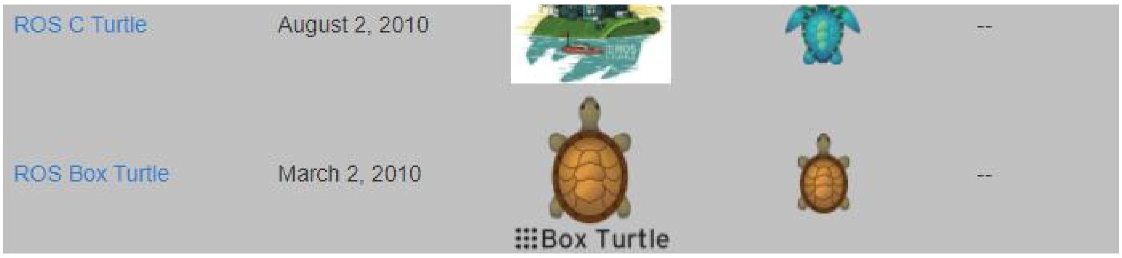
7.1.4 Common components

launch file; TF coordinate transformation; Rviz; Gazebo; QT toolbox; Navigation; MoveIt!

7.1.5 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	Box Turtle		--

A ROS distribution is a versioned set of ROS packages. These are akin to Linux distributions (e.g. Ubuntu). The purpose of the ROS distributions is to let developers work against a relatively stable codebase until they are ready to roll everything forward. Therefore once a distribution is released, we try to limit changes to bug fixes and non-breaking improvements for the core packages (every thing under `ros-desktop-full`). And generally that applies to the whole community, but for "higher" level packages, the rules are less strict, and so it falls to the maintainers of a given package to avoid breaking changes. As of October 2019, the version name, release time and version life cycle of the main release version of ROS are shown in the following table:

Version name	**Date**	**Life cycle**	**Operating system platform**
ROS Noetic Ninjemys	May 2020	May 2025	Ubuntu 20.04
ROS Melodic Morenia	23th May 2018	May 2023	Ubuntu 17.10, Ubuntu 18.04, Debian 9, Windows 10
ROS Lunar Loggerhead	23th May 2017	May 2019	Ubuntu 16.04, Ubuntu 16.10, Ubuntu 17.04,Debian 9
ROS Kinetic Kame	23th May 2016	April 2021	Ubuntu 15.10, Ubuntu 16.04, Debian 8
ROS Jade Turtle	23th May 2015	May 2017	Ubuntu 14.04, Ubuntu 14.10, Ubuntu 15.04
ROS Indigo Igloo	22th July 2014	April 2019	Ubuntu 13.04, Ubuntu 14.04
ROS Hydro Medusa	4th September 2014	May 2015	Ubuntu 12.04, Ubuntu 12.10, Ubuntu 13.04
ROS Groovy Galapagos	31th December 2013	July 2014	Ubuntu 11.10, Ubuntu 12.04, Ubuntu 12.10
ROS Fuerte Turtle	23th April 2012	--	Ubuntu 10.04, Ubuntu 11.10, Ubuntu 12.04
ROS Electric Emys	30th August 2011	--	Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04, Ubuntu 11.10
ROS Diamondback	2nd March 2010	--	Ubuntu 10.04, Ubuntu 10.10, Ubuntu 11.04
ROS C Turtle	2nd August 2010	--	Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04, Ubuntu 10.10
ROS Box Turtle	2nd March 2010	--	Ubuntu 8.04, Ubuntu 9.04, Ubuntu 9.10, Ubuntu 10.04