15.About ROS2 meta-function package

1. Introduction to meta-function package

Completing a systematic function may involve multiple functional packages. For example, the robot navigation module is implemented. Under this module, there are different sub-level functional packages such as map, positioning, path planning, etc. So when the caller installs the module, does it need to install each function package one by one?

Obviously, installing function packages one by one is inefficient. In ROS2, a way is provided to package different function packages into one function package. When installing a certain function module, just call the packaged function package directly. This package is also called a metapackage.

MetaPackage is a file management system concept in Linux. It is a fake package in ROS2, with no substantive content in it.But it depends on other software packages. In this way, other packages can be combined. We can think of it as a directory index of a book, telling us which sub-packages are in this package collection and where to download it.

For example:

The sudo apt install ros-foxy-desktop command uses the meta-function package when installing ros2. This meta-function package depends on some other function packages in ROS2. When installing this package, the dependencies will be installed together.

2. Function

To facilitate user installation, we only need this one package to organize and install other related software packages together.

3. Realize

1. Create a new function package

```
ros2 pkg create pkg_metapackage
```

2. Modify the package.xml file and add the packages that execution depends on.

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>pkg_metapackage</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="1461190907@qq.com">root</maintainer>
```

```
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <exec_depend>pkg_interfaces</exec_depend>
  <exec_depend>pkg_helloworld_py</exec_depend>
  <exec_depend>pkg_topic</exec_depend>
  <exec_depend>pkg_service</exec_depend>
  <exec_depend>pkg_action</exec_depend>
  <exec_depend>pkg_param</exec_depend>

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

3. The content of the file CMakeLists.txt is as follows

```
cmake_minimum_required(VERSION 3.5)
project(pkg_metapackage)

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

find_package(ament_cmake REQUIRED)

ament_package()
```