

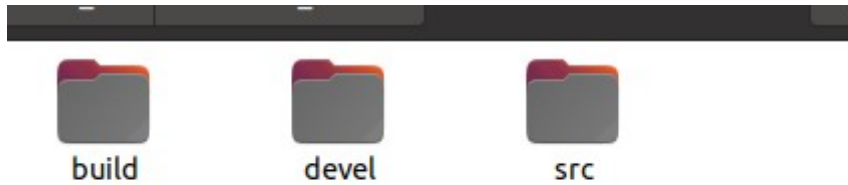
在 Vscode 下对 Ros 下进行编译

1、`mkdir -p demo02_ws/src`

创建一个工作空间 `demo02_ws` 下面有个 `src` 文件

2、`cd demo02_ws`

进入工作空间 `demo02_ws`



3、`catkin_make`

创建一个 `ros` 环境

3、`code .`

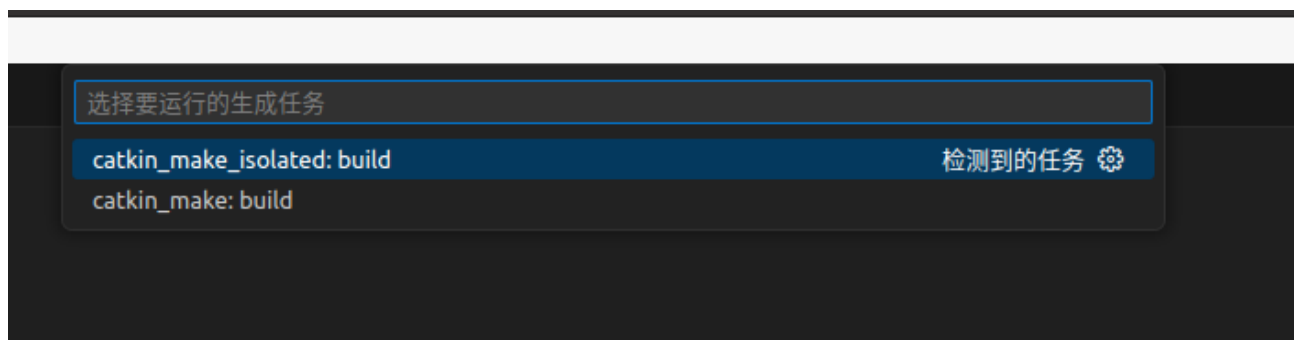
`code` + 空格 + `.` 点

打开了 `vscode`

4、`ctrl shift b`

去设置编译器

选择 `catkin_make::build` 之后他的右边也会出现一个设置的小按钮 点击以下



效果如下

```

{
  "version": "2.0.0",
  "tasks": [
    {
      "type": "catkin_make",
      "args": [
        "--directory",
        "/home/qinghuan/env_cv/demo02_ws",
        "-DCMAKE_BUILD_TYPE=RelWithDebInfo"
      ],
      "problemMatcher": [
        "$catkin-gcc"
      ],
      "group": "build",
      "label": "catkin_make: build"
    }
  ]
}

```

这个是 task 文件的内容 我们要基于模板天津内容

```

{
  // 有关 tasks.json 格式的文档，请参见
  // https://go.microsoft.com/fwlink/?LinkId=733558
  "version": "2.0.0",
  "tasks": [
    {
      "label": "catkin_make:debug", //代表提示的描述性信息
      "type": "shell", //可以选择 shell 或者 process,如果是 shell 代码是在 shell 里面运行一个命令，如
      //果是 process 代表作为一个进程来运行
      "command": "catkin_make", //这个是我们需要运行的命令
      "args": [], //如果需要在命令后面加一些后缀，可以写在这里，比如-
      //DCATKIN_WHITELIST_PACKAGES="pac1;pac2"
      "group": {"kind": "build", "isDefault": true},
      "presentation": {
        "reveal": "always" //可选 always 或者 silence，代表是否输出信息
      },
      "problemMatcher": "$msCompile"
    }
  ]
}

```

可以点击配置设置为默认，修改.vscode/tasks.json 文件

```
{
  // 有关 tasks.json 格式的文档，请参见
  // https://go.microsoft.com/fwlink/?LinkId=733558
  "version": "2.0.0",
  "tasks": [
    {
      "label": "catkin_make:debug", //代表提示的描述性信息
      "type": "shell", //可以选择shell或者process,如果是shell代码是在shell里面运行一个命令,
      "command": "catkin_make",//这个是我们需要运行的命令
      "args": [],//如果需要在命令后面加一些后缀,可以写在这里,比如-DCATKIN_WHITELIST_PACKAGE
      "group": {"kind": "build", "isDefault": true},
      "presentation": {
        "reveal": "always"//可选always或者silence,代表是否输出信息
      },
      "problemMatcher": "$msCompile"
    }
  ]
}
```

截图的效果 如图所示

如果我们的配置文件这么写 那么在输入
Ctrl + shift + b 就自动编译这个空间了

5、在右侧 src 文件夹 地方右键 可以看到创建功能包
之后 设置你功能包的名字
设置你的 ros 文件的依赖文件
hello_world
这是我的功能包的名字
roscpp rospy std_msgs
是 c++ python 编译语言依赖
std_msgs 是 ros 的通信机制



如果害怕功能包 出现问题 ctrl shit b 来编译看是否存在问题

有问题会直接报错的

6、在 vscode 的 hello_vscode 的文件夹下 开始编译自己的 c++文件

```

src > hello_vscode > src > G+ hello_vscode_c.cpp > main(int, char *[])
1  #include "ros/ros.h"
2  // 修改了 cpp文件下 c++14变成了c++17
3  /*
4  "cppStandard": "c++17" 这一项
5  对应的是cppStandard
6
7  */
8  int main(int argc, char *argv[])
9  {
10     ros::init(argc, argv, "hello");
11     ROS_INFO("hahaha");
12     ros::spin();
13     return 0;
14 }

```

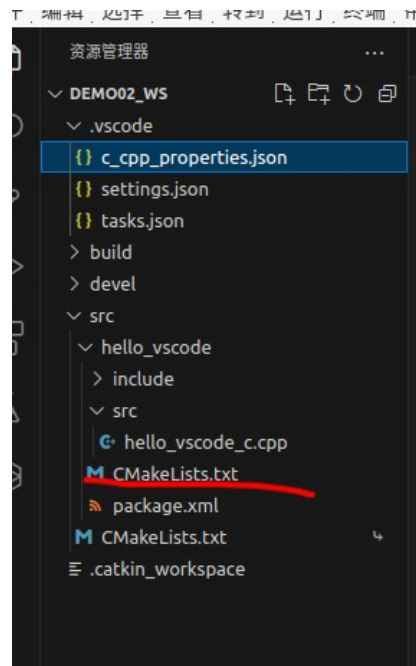
cpp 对应的 json 文件 改红线的位置 变为 17 就不会有 ros 的红线显示 没有这个库 这个c_cpp_properties.json

```

de > {} c_cpp_properties.json > [ ] configurations > {} 0 > [ ] cppStandard
{
  "configurations": [
    {
      "browse": {
        "databaseFilename": "${default}",
        "limitSymbolsToIncludedHeaders": false
      },
      "includePath": [
        "/opt/ros/noetic/include/**",
        "/usr/include/**"
      ],
      "name": "ROS",
      "intelliSenseMode": "gcc-x64",
      "compilerPath": "/usr/bin/gcc",
      "cStandard": "gnu11",
      "cppStandard": "c++17"
    }
  ],
  "version": 4
}

```

6、更改配置文件 CMakeLists.txt



136 149 两行 和原本的文件操作是一样的

```
117 ## Your package locations should be listed before other locations
118 include_directories(
119   # include
120   ${catkin_INCLUDE_DIRS}
121 )
122
123 ## Declare a C++ library
124 # add_library(${PROJECT_NAME}
125 #   src/${PROJECT_NAME}/hello_vscode.cpp
126 # )
127
128 ## Add cmake target dependencies of the library
129 ## as an example, code may need to be generated before libraries
130 ## either from message generation or dynamic reconfigure
131 # add_dependencies(${PROJECT_NAME} ${${PROJECT_NAME}_EXPORTED_TARGETS} ${c
132
133 ## Declare a C++ executable
134 ## With catkin_make all packages are built within a single CMake context
135 ## The recommended prefix ensures that target names across packages don't
136 add_executable(hello_vscode_c src/hello_vscode_c.cpp)
137
138 ## Rename C++ executable without prefix
139 ## The above recommended prefix causes long target names, the following re
140 ## target back to the shorter version for ease of user use
141 ## e.g. "roslaunch someones_pkg node" instead of "roslaunch someones_pkg someo
142 # set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME node P
143
144 ## Add cmake target dependencies of the executable
145 ## same as for the library above
146 # add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS
147
148 ## Specify libraries to link a library or executable target against
149 target_link_libraries(hello_vscode_c
150   ${catkin_LIBRARIES}
151 )
152
153 #####
```

映射 136 第一个参数意义 第二个是我们cpp 文件的名字

第一个参数是 对应136行的映射

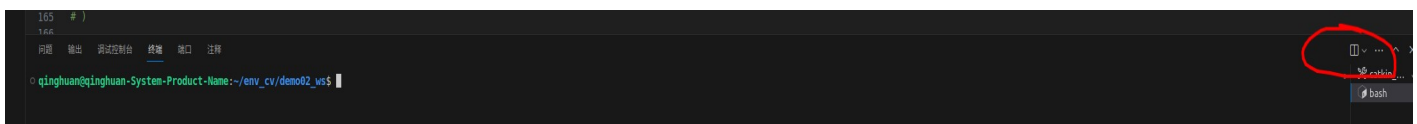
之后通过 `ctrl + shift + b` 来编译一下
没有问题的编译结果



```
问题 输出 调试控制台 终端 端口 注释

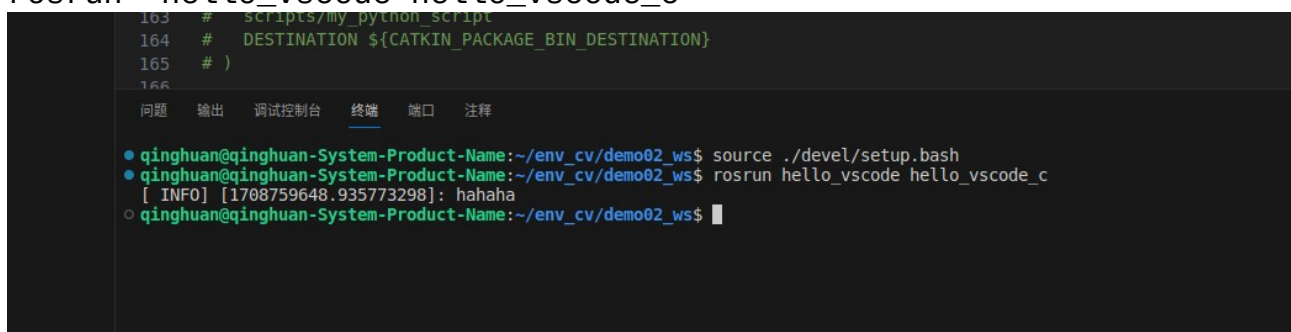
-- ==> add_subdirectory(hello_vscode)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/qinghuan/env_cv/demo02_ws/build
####
#### Running command: "make -j12 -l12" in "/home/qinghuan/env_cv/demo02_ws/build"
####
Scanning dependencies of target hello_vscode_c
[ 50%] Building CXX object hello_vscode/CMakeFiles/hello_vscode_c.dir/src/hello_vscode_c.cpp.o
[100%] Linking CXX executable /home/qinghuan/env_cv/demo02_ws/devel/lib/hello_vscode/hello_vscode_c
[100%] Built target hello_vscode_c
* 终端将被任务重用，按任意键关闭。
```

红线的位置是一个+ 号 可以看一个新的终端 这个终端直接定位到了 包的工作路径



`roscore` 运行 `ros` 核心

在新开一个终端 通过右侧的加号
`source ./devel/setup.bash`
`roslaunch hello_vscode hello_vscode_c`



```
163 # scripts/my_python_script
164 # DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
165 # )
166
问题 输出 调试控制台 终端 端口 注释

● qinghuan@qinghuan-System-Product-Name:~/env_cv/demo02_ws$ source ./devel/setup.bash
● qinghuan@qinghuan-System-Product-Name:~/env_cv/demo02_ws$ roslaunch hello_vscode hello_vscode_c
[ INFO] [1708759648.935773298]: hahaha
○ qinghuan@qinghuan-System-Product-Name:~/env_cv/demo02_ws$
```

之后调用了最后的结果 就算 `hahahah` 就算 `ROS_INFO` 的方法

创建功能包

4.4 创建 ROS 功能包

选定 src 右击 ---> create catkin package

设置包名 添加依赖

代码没有提示 以及中文

PS1: 如果没有代码提示

修改 .vscode/c_cpp_properties.json

设置 "cppStandard": "c++17"

PS2: main 函数的参数不可以被 const 修饰

PS3: 当ROS__INFO 终端输出有中文时, 会出现乱码

INFO: ??????????????????????????????

解决办法: 在函数开头加入下面代码的任意一句

```
setlocale(LC_CTYPE, "zh_CN.utf8");  
setlocale(LC_ALL, "");
```

Copy

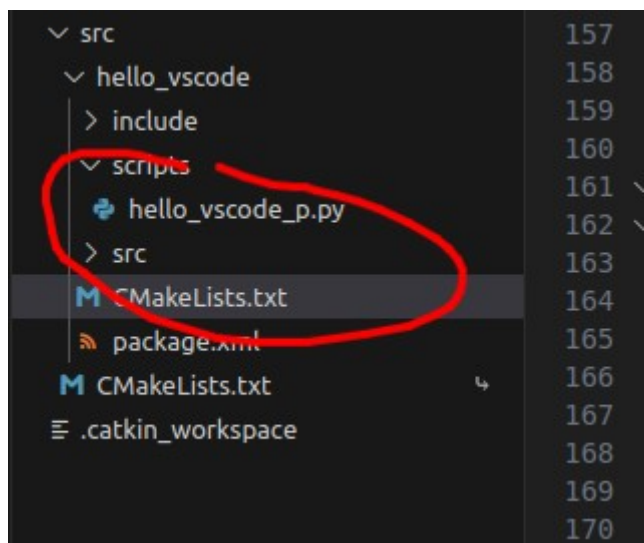
解决中文乱码方法

```
> hello_vscode > src > G+ hello_vscode_c.cpp > main(int, char * [])  
1  #include "ros/ros.h"  
2  // 修改了 cpp文件下 c++14变成了c++17  
3  /*  
4  "cppStandard": "c++17" 这一项  
5  对应的是cppStandard  
6  
7  */  
8  int main(int argc, char *argv[])  
9  {  
10     setlocale(LC_CTYPE, "zh_CN.utf8");  
11     ros::init(argc, argv, "hello");  
12  
13     ROS_INFO("hahaha");  
14     ROS_INFO("你好scode");  
15     return 0;  
16 }
```

python 在 vscode 的编译方法

创建一个 scripts 文件

这个是 hello_vscode 文件夹下和 src 同级

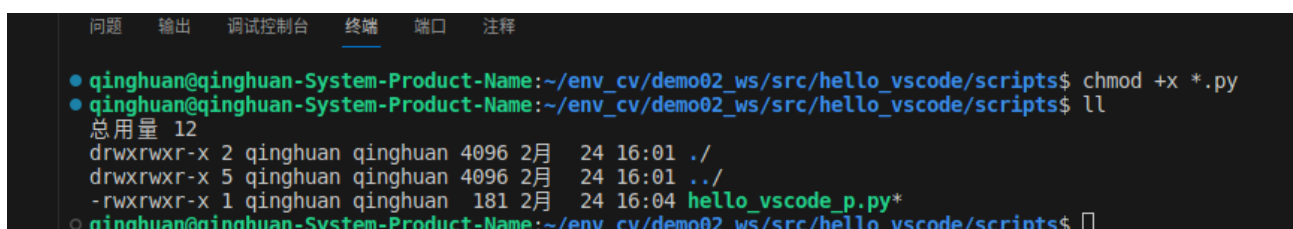


创建一个 python 文件

```
src > hello_vscode > scripts > + hello_vscode_p.py
1  #!/user/bin/env python
2  #"""声明了一个解释器"""
3
4  import rospy
5
6  if __name__ == '__main__':
7      rospy.init_node("hello-python")
8      rospy.loginfo("hello world 这是python")
```

之后对 scripts 文件右键 选择在终端打开

chmod +x *.py 给所有 py 文件添加可以执行的权限



之后修改 cmake 文件

和之前在终端的 python 方法类似
修改需要执行的文件

```
159
160  ## Mark executable scripts (Python etc.) for installation
161  ## in contrast to setup.py, you can choose the destination
162  catkin_install_python(PROGRAMS
163    scripts/hello_vscode_p.py
164    DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
165  )
166
167  ## Mark executables for installation
168  ## See http://docs.ros.org/melodic/api/catkin/html/howto/format1/building_executables.html
169  # install(TARGETS ${PROJECT_NAME}_node
```

之后编译

ctrl shift + B

编译文件 通过上述命令

在一个终端 roscore 启用 ros

source ./devel/setup.bash

更新环境便利

roslaunch hello_vscode hello_vscode_p.py

运行 hello_vscode 功能包的 hello_vscode_p.py 文件

```
问题 输出 调试控制台 终端 端口 注释
• qinghuan@qinghuan-System-Product-Name:~/env_cv/demo02_ws$ source ./devel/setup.bash
• qinghuan@qinghuan-System-Product-Name:~/env_cv/demo02_ws$ roslaunch hello_vscode hello_vscode_p.py
/home/qinghuan/env_cv/demo02_ws/src/hello_vscode/scripts/hello_vscode_p.py:7: UserWarning: 'hello-python' is not a legal ROS base name. This may cause problems with other ROS tools.
  rospy.init_node('hello-python')
[INFO] [1708762440.546549]: hello world 这是python
• qinghuan@qinghuan-System-Product-Name:~/env_cv/demo02_ws$
```