ubuntu vscode配置c++环境

使用ubuntu vscode配置c++开发环境

1. 安装c/c++的插件

在vscode中搜索c++安装c/c++的插件.

2. 建立工程

在c++中一个工程是以文件夹的形式进行管理的, 因此建立一个工程就是新建一个文件夹, 这里新建工程

```
mkdir hello
```

3. 更改配置文件launch.json

在vscode中打开这个文件夹,新建一个文件为main.cpp,然后输入下列测试程序并保存:

```
#include<iostream>
int main(){
    std::cout<<"hello world! "<<std::endl;
    return 0;
}
```

点击vscode左侧的debug按钮(小虫子), 选择添加配置(Add configuration), 选择c++(GDB/LIDB), 然后将会自动生成launch.json文件.

默认的json文件为:

```
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?
linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(gdb) Launch",
            "type": "cppdbg",
            "request": "launch",
            "program": "enter program name, for example
${workspaceFolder}/a.out",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceFolder}",
            "environment": [],
            "externalConsole": true,
            "MIMode": "gdb",
            "preLaunchTask"="build",
            "setupCommands": [
                {
                    "description": "Enable pretty-printing for gdb",
```

```
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                }
            ]
        }
    ]
```

然后需要修改其中的部分文档,将program项目的内容更改为调试时运行的程序:

```
 "program": "${workspaceFolder}/main.out",
```

4. 添加构建(编译\链接等)任务(tasks.json)

为了方便在VScode里编译C++代码，我们可以将类似`g++ -g main.cpp`等g++命令写入VScode的任务系统。
首先，利用快捷键ctrl+shift+p打开命令行，输入`Tasks: Run task`，会出现如下提示：

```
No task to run found. configure tasks...
```

回车，然后依次选择如下：

```
 Create tasks.json file from template

 OthersExample to run an arbitrary external command.
```

生成默认的tasks.json文件

```
{
    // See https://go.microsoft.com/fwlink/?LinkId=733558
    // for the documentation about the tasks.json format
    "version": "2.0.0",
    "tasks": [
        {
            "label": "echo",
            "type": "shell",
            "command": "echo Hello"
        }
    ]
}
```

这里的`label`为任务名，我们将`"label"= "echo"`改为`"label"= "build"`。
由于我们的指令是`g++`，这里将`"command"="echo Hello"`改为`"command"="g++"`。
然后添加`g++`的参数`args`。如果我们的g++指令为：`g++ -g main.cpp`，这里可以把参数设置为如下：

```
{
    "tasks": [
        {
            "label": "build",
            "type": "shell",
            "command": "g++",
            "args": ["-g", "${file}"]
        }
      ]
}
```

如果我们想配置g++指令为：`g++ -g main.cpp -std=c++11 -o main.out`，则参数可设置为：

```
{
    "tasks": [
        {
            "label": "build",
            "type": "shell",
            "command": "g++",
            "args": ["-g", "${file}", "-std=c++11", "-o",
"${fileBasenameNoExtension}.out"]
        }
     ]
}
```

我们可以通过举一反三来配置不同的g++指令。完整的tasks.json文件可参考附录。

5. 断点调试

经过上述配置之后就可以对我们写的程序进行简单的配置。在进行下面的操作前，我们应当保证`launch.json`和`tasks.json`的正确性并且已经成功保存。

使用快捷键ctrl+shift+p调出命令行，选择执行我们的`build`任务，build成功后，点击开始调试.

注: 每次新建工程之后都得进行这一系列的操作才可以进行断点调试的相关调试.

附注 g++命令参数

```
-o：指定生成可执行文件的名称。使用方法为：g++ -o afile file.cpp file.h ... （可执行文件不可与待编译或链接文件同名，否则会生　　成相应可执行文件且覆盖原编译或链接文件），如果不使用-o选项，则会生成默认可执行文件a.out。


-c：只编译不链接，只生成目标文件。

-g：添加gdb调试选项。
```