

Clipped from: http://www.yalewoo.com/sublime_text_3_gcc.html

sublime text 3 提供了构建功能，它的构建系统 (Build systems) 可以运行一段外部命令，还可以捕获输出并显示。

要在 sublime text 3 中实现 c 或 c++ 代码的编译和运行，在本质上说也是**调用外部的命令**，windows 中也可以理解为执行一段 cmd 命令。

目前 c/c++ 编译器最流行的就是 gcc 和 g++，本文将从 MinGW 开始，介绍 gcc 和 g++ 的基本命令格式，然后详细介绍 sublime 中自带的编译配置文件，分析每一行的作用。然后给出 win7 64bit 下 Sublime Text 3 build 3083 版本中编译 c 语言、c++ 的 build 配置文件。

另外，文章最后还介绍了 sublime 中使用 make 的内容，以及讨论关于中文编码的问题。

如果你只想快速配置好编译环境，而对实现细节并不关心（**不建议**），你可以只阅读 安装 MinGW、配置环境变量 和 编写自己的编译配置文件这几节。

关于 gcc 和 g++

安装编译器是后面所有工作的基础，如果没有编译器，后面的一切都无从谈起。在 windows 下使用 gcc 和 g++，是通过安装 MinGW 实现的。

安装 MinGW

MinGW 是 Minimalist GNU on Windows 的首字母缩写，安装后就可以使用很多的 GNU 工具。GNU (GNU's Not Unix) 是 linux 中的一个著名的项目，包含了 gcc\g++\gdb 等工具。也就是说，安装 MinGw 后，我们就可以使用 gcc 和 g++ 命令了。

MinGW 的官网是 <http://www.mingw.org/>，但是从官网安装很麻烦，在线安装经常龟速容易失败。

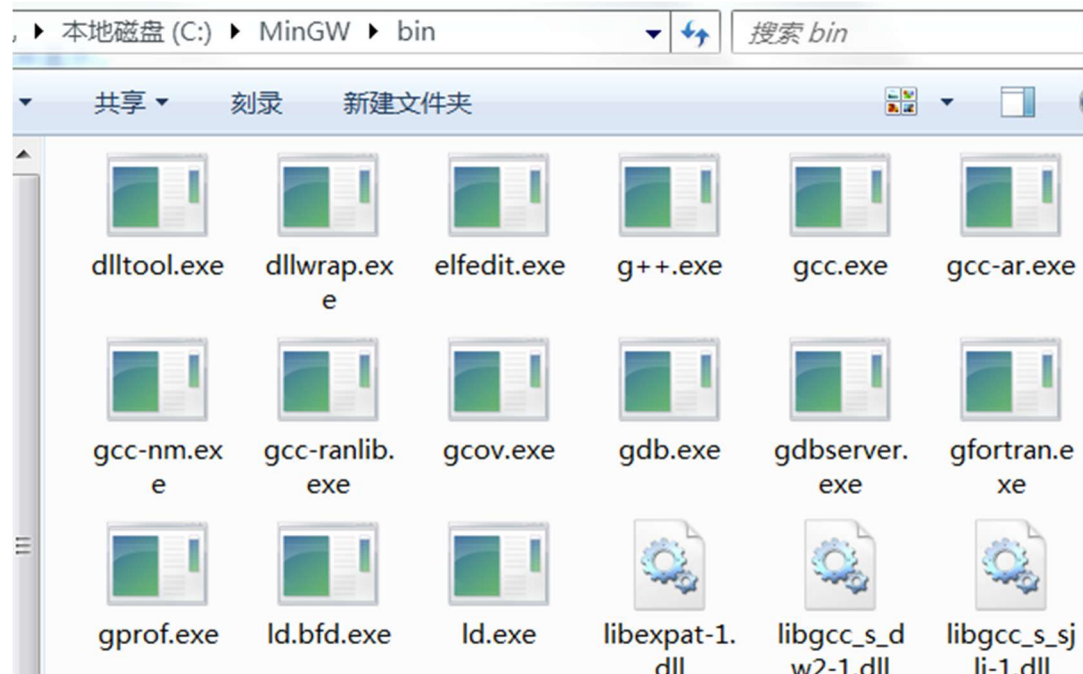
雅乐网推荐的方法是借助 [codeblocks](#)，选择带有 mingw 的版本安装，安装后把 mingw 文件夹复制出来就可以了。

这里提供了解压版的 MinGW，是使用 codeblocks-13.12mingw-setup 安装后复制出来的：

<http://pan.baidu.com/s/1gd5YzVP>

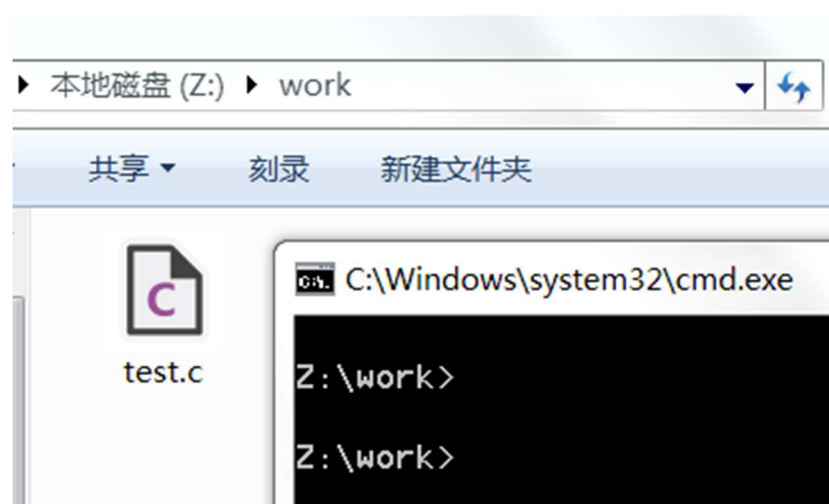
解压后，可以在 MinGW/bin 目录下找到我们需要的 gcc.exe 和 g++.exe。

我这里把 MinGW 文件夹放到 c 盘根目录



在cmd 中使用gcc

假设我们有一个 test.c 文件在 Z 盘的 work 目录下。首先我们要在 cmd 中进入此目录。方法可以是在 work 目录空白处按住 Shift 点击鼠标右键，选择“在此处打开命令窗口”；也可以[使用 cd 命令](#)进入。



gcc 的一般格式是

```
1 gcc 源文件名 -o 可执行文件名
```

但是我们输入命令

```
1 gcc test.c -o test
```

执行后却提示

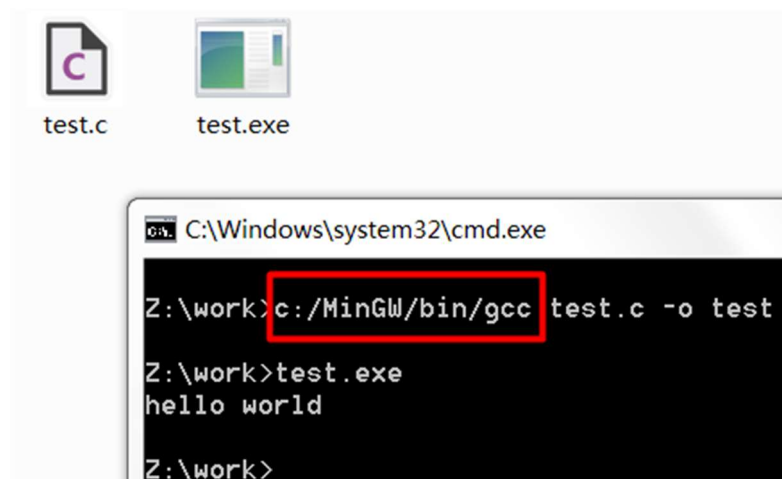
'gcc' 不是内部或外部命令，也不是可运行的程序或批处理文件。

这是因为命令执行时，会在当前目录下查找名为 gcc 的可执行文件，如果查不到就在系统环境变量 path 记录的路径里寻找 gcc 可执行文件。但是目前这两个地方都没有。我们的 gcc 文件所在的目录是 c 盘下的 MinGW/bin。

这时可以使用绝对路径来调用 gcc 可执行文件

```
1 Z:\work>c:/MinGW/bin/gcc test.c -o test
2
3 Z:\work>test.exe
4 hello world
```

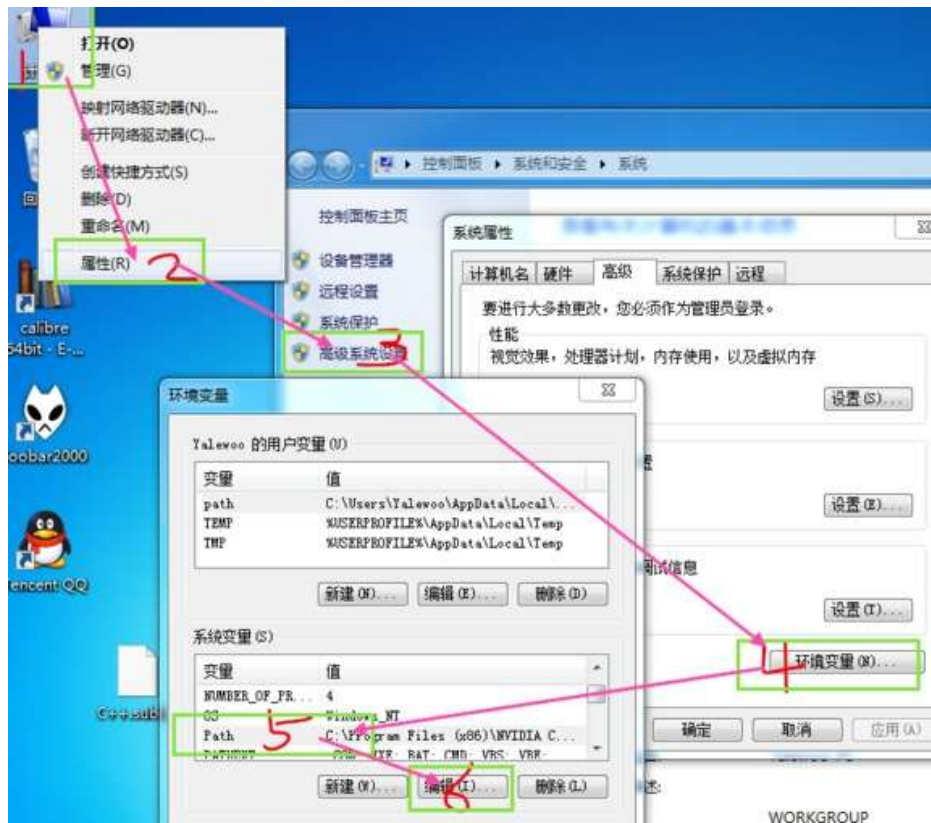
这样就成功编译生成了可执行文件 test.exe，然后就可以在 cmd 里运行了。



配置环境变量

为了方便，一般我们会把 gcc 所在的路径加入系统的环境变量，这样就可以直接使用 gcc 命令而不用绝对路径。

右键计算机 -> 属性 -> 高级系统设置 -> 环境变量



在 path 的值中，可以发现有一些目录，他们之间用**英文的分号**分隔。我们双击 path，把我们 gcc 的路径 C:\MinGW\bin 添加进去。 要注意前后的**英文分号**。



确定以后 就可以在任意目录下直接使用 gcc 命令了。可以在任意目录打开 cmd 窗口，输入 gcc 查看环境变量是否设置成功。如果仍然提示不是内部或外部命令，说明环境变量设置失败。



注意：在 sublime text 3 build 3083 中，环境变量的修改不会立即在 sublime 中生效，需要重启 windows。

[cmd 编译运行 c 语言](#)

总结一下流程：

首先我们要在 cmd 中进入 .c 文件所在的目录作为工作目录

然后执行 `gcc source.c -o dest` 来生成可执行文件

最后输入生成的可执行文件名来运行生成的程序。

建议大家加入 `-Wall` 选项，打开常用的警告。

下面是几种常用的命令：

编译 c 语言

```
1 gcc -Wall 源文件名 -o 可执行文件名
```

编译 c++ 语言

```
1 g++ -Wall 源文件名 -o 可执行文件名
```

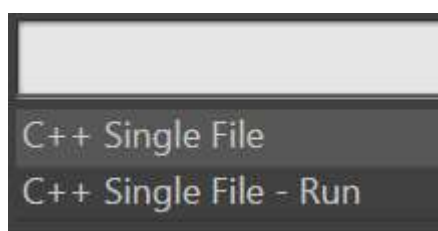
调试 c++

```
1 g++ -g 源文件名 -o 可执行文件名
2 gdb 可执行文件名
```

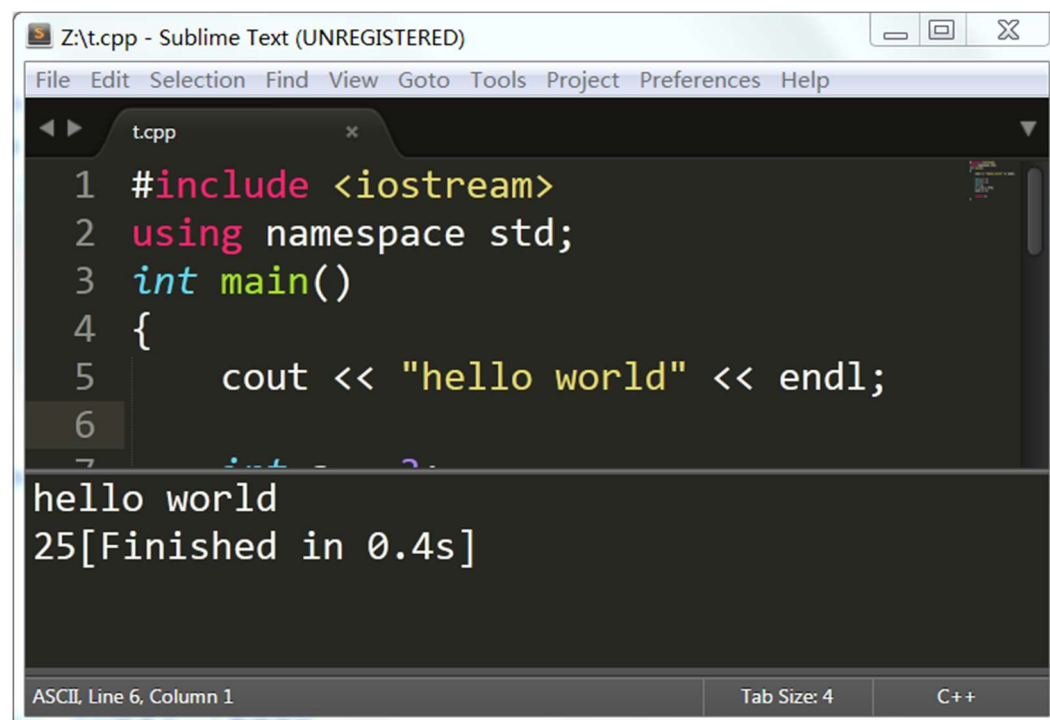
[Sublime Text 3 默认 c/c++ 编译配置文件详解](#)

把 `g++` 加入环境变量后，sublime 中默认的编译系统就可以正常使用了。

我们在 Sublime Text 3 中打开一个 `cpp` 文件，按 `Ctrl+B`



这是 sublime 自带的默认 c++ 编译命令。第一个是编译，第二个是运行。这时候是可以正常使用的。（环境变量配置后需重启 windows）



The screenshot shows the Sublime Text 3 interface with a file named 't.cpp' open. The code in the editor is as follows:

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     cout << "hello world" << endl;
6 }
```

Below the code editor, the output panel displays the result of running the program:

```
hello world
25[Finished in 0.4s]
```

The status bar at the bottom indicates 'ASCII, Line 6, Column 1', 'Tab Size: 4', and 'C++'.

Sublime Text 3 3080 版本之后修改了编译系统，具体设置是

Ctrl+B 执行改格式上次的编译命令。如果第一次执行则提示选择执行哪个

Ctrl+Shift+B 选择执行哪个

不足之处：

1. 程序输出捕获到 Sublime 窗口中，这样导致不能运行时输入信息。执行含有 scanf 语句的代码会卡住。
2. 默认情况下 c 和 c++ 没有进行区分，全部当做 c++ 格式来处理了。

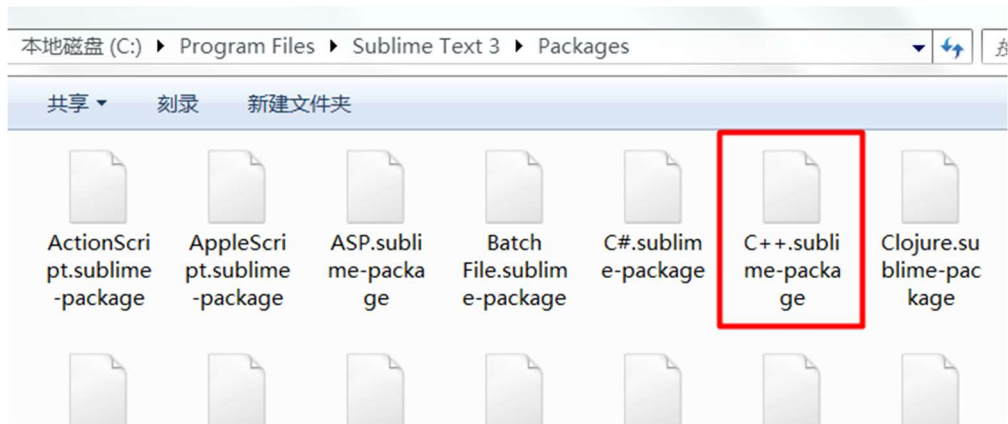
解决办法

第一个是设置在新的 cmd 窗口执行程序，这样就可以输入信息。

第二个是针对 c 语言单独写一个 build 配置文件。

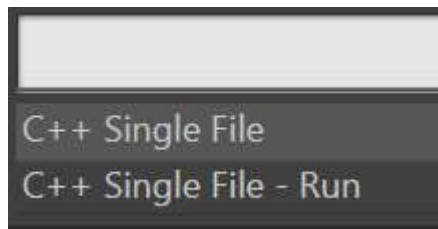
默认的编译配置文件位置

在 Sublime 的安装目录的 Packages 文件夹中，有个文件叫 C++.sublime-package



这个实际上是 zip 的压缩包，里面包含了 c++ 的默认系统设置，**修改后缀名为 zip 后解压**，可以在里面找到 **C++ Single File.sublime-build** 文件。

这个文件名就是上面要编译时可以选择的名字



[编译配置文件详解](#)

这个文件内容如下：

```
1      {
2      "shell_cmd": "g++ \"${file}\"-o
3      \"${file_path}/${file_base_name}\",
4      "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)??:? (.*)$",
5      "working_dir": "${file_path}",
6      "selector": "source.c, source.c++",
7
8      "variants":
9      [
10     {
11     "name": "Run",
12     "shell_cmd": "g++ \"${file}\"-o
13     \"${file_path}/${file_base_name}\"&&
14     \"${file_path}/${file_base_name}\"
15     }
16     ]
17     }
```

这个 JSON 格式的配置文件就是 sublime 中 build 文件的真面目了。花括号里面是一个个的键值对，它们之间用逗号隔开。键和值中间是一个冒号。为了方便下面把键称为名称。

名称和值都要用双引号括起来，因此值里面用到双引号的话，就要用转义 \"（反斜杠 + 双引号）表示。

这里面用到的名称的含义如下：

名称 含义

working_dir	运行 cmd 是会先切换到 working_dir 指定的工作目录
cmd	包括命令及其参数。如果不指定绝对路径，外部程序会在你系统的: const:PATH 环境变量中搜索。
shell_cmd	相当于 shell:true 的 cmd，cmd 可以通过 shell 运行。
file_regex	该选项用 Perl 的正则表达式来捕获构建系统的错误输出到 sublime 的窗口。
selector	在选定 Tools Build System Automatic 时根据这个自动选择编译系统。
variants	用来替代主构建系统的备选。也就是一个配置文件可以对应多个执行命令
name	只在 variants 下面有，设置命令的名称，例如 Run。

上面的配置文件中还有一些类似 \${file} 这种符号，这是 sublime 提供的变量，一些常用的变量如下：

变量 含义

\$file_path	当前文件所在目录路径, e.g., <i>C:\Files</i> .
\$file	当前文件的详细路径, e.g., <i>C:\Files\Chapter1.txt</i> .
\$file_name	文件全名（含扩展名）, e.g., <i>Chapter1.txt</i> .
\$file_extension	当前文件扩展名, e.g., <i>txt</i> .
\$file_base_name	当前文件名（不包括扩展名）, e.g., <i>Document</i> .

变量的使用可以直接使用，也可以使用花括号括起来，例如 `${project_name}`

我们详细看一下这个文件。

```
1      "working_dir": "${file_path}",
```

这一行说明工作目录，也就是执行命令时所在的目录，被设置为文件所在的目录。

下面的

```
1      "shell_cmd": "g++ \"${file}\"-o  
      \"${file_path}/${file_base_name}\"",
```

这就是编译时执行的命令了，它的值的部分是

```
1      g++ \"${file}\" -o \"${file_path}/${file_base_name}\"
```

执行时，`${file}` 会被替换为编辑的文件名，`${file_path}/${file_base_name}` 就会被替换为不包含扩展名的完整路径名。它们前后有 `\` 双引号是为了支持带空格的文件名。

假设我们编辑的文件路径为 `Z:/cpp/t.cpp`，那么执行时的工作目录就是 `Z:/cpp`。

`${file}` 就是 `Z:/cpp/t.cpp`，`${file_path}` 就是 `Z:/cpp`，`${file_base_name}` 就是 `t`。

我们把转义字符双引号也用双引号表示，这样执行的命令就变成了

```
1      g++ "Z:/cpp/t.cpp" -o "Z:/cpp/t"
```

这正好是编译文件的命令，文件路径前后的双引号保证它支持带空格的路径。编译后生成的可执行文件和源代码文件在一个目录下，名字相同（扩展名不同）。

如果有编译错误，错误信息就会被“`file_regex`”中的正则表达式匹配并显示。

再来看最后面的代码

```
1      "variants":  
2      [  
3      {  
4      "name": "Run",  
5      "shell_cmd": "g++ \"${file}\"-o  
6      \"${file_path}/${file_base_name}\"&&  
7      \"${file_path}/${file_base_name}\""
```

```
}  
]
```

variants 的值是一个数组，可以放很多个对象，每个对象表示一个命令。里面 name 表示了这个命令的名称为 Run，也就是运行。编译时选择 C++ Single File-Run 就会执行这里的 shell_cmd。

运行部分的命令前半部分和编译一样，后面用 && 连接了另一个命令，&& 表示编译成功才执行后面的部分。后面为

```
1      \"${file_path}/${file_base_name}\"
```

也就是直接运行可执行文件了。这样是在 sublime 中捕获运行结果的。

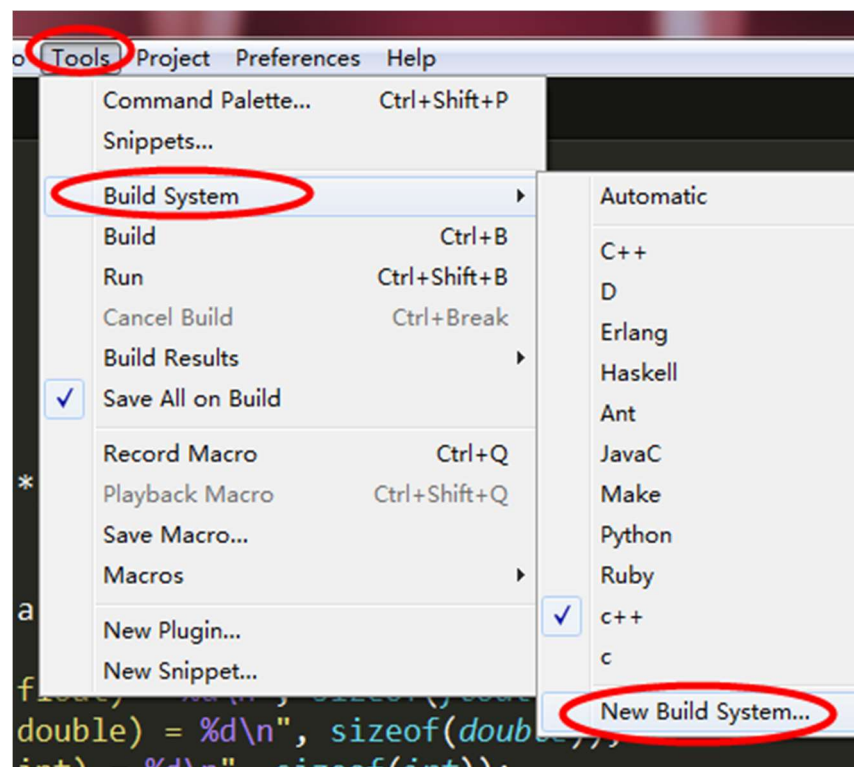
看懂这个默认配置文件后，照葫芦画瓢，我们很容易的就可以写出符合自己需要的配置文件。

但是不建议直接修改这个文件，建议大家把用户配置放到用户文件夹下，来代替默认的编译配置。

编写自己的编译配置文件

c 语言

选择 tool -> Build System -> New Build System



然后输入以下代码

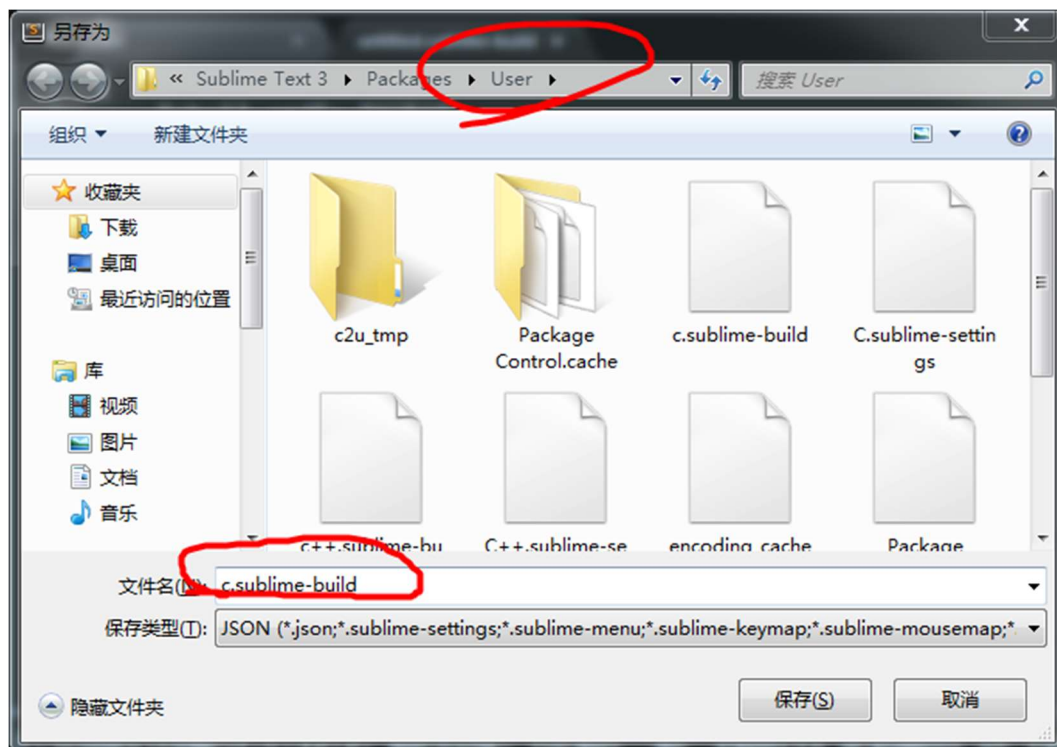
```

1      {
2      "cmd": ["g++", "${file}", "-o", "${file_path}/${file_base_name}"],
3      "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)?(?:.*)$",
4      "working_dir": "${file_path}",
5      "encoding": "cp936",
6      "selector": "source.c",
7      "variants":
8      [
9      {
10     {
11     "name": "Run",
12     "cmd": ["cmd", "/C", "start", "cmd", "/c",
13     "${file_path}/${file_base_name}.exe & pause"]
14     }
15     }
16     ]
17     }

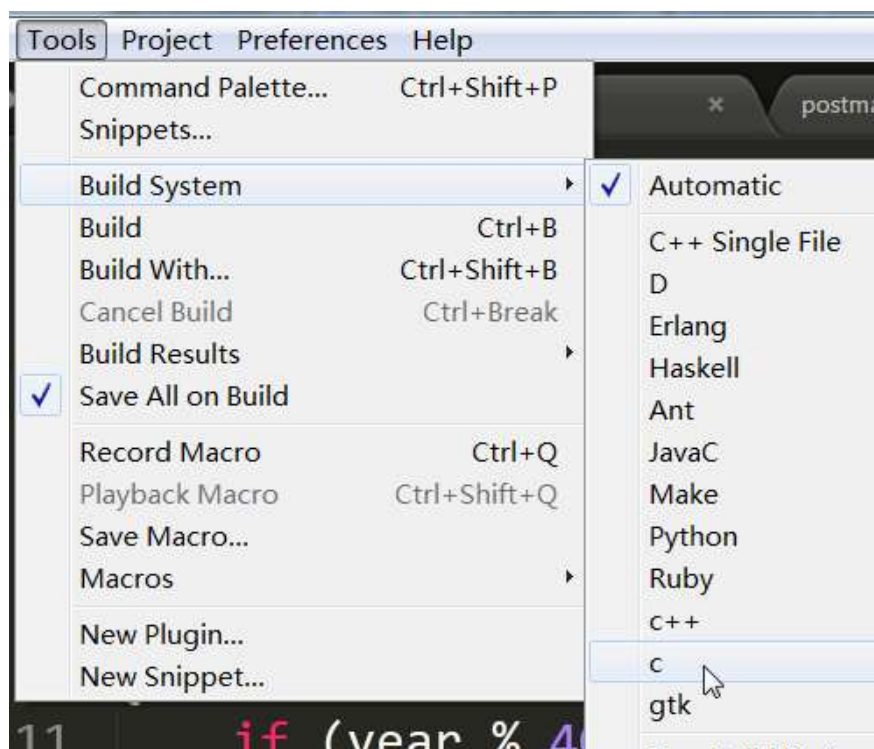
```

和默认相比，就是修改了 selector 部分为只选择.c 文件。**Run** 中的 shell_cmd 后面部分加上了 start cmd /c，start 作用是新开一个 cmd 窗口，cmd 表示要执行一个命令行，/c 执行完后退出新开的窗口，后面的 & pause 保证运行结束后窗口不会立即退出。这样 Run 就会在新的 cmd 窗口中运行了。

按 Ctrl+s 保存，会自动打开 user 目录（Sublime Text 3\ Packages\User），我们修改文件名为 c.sublime-build，保存在此目录。



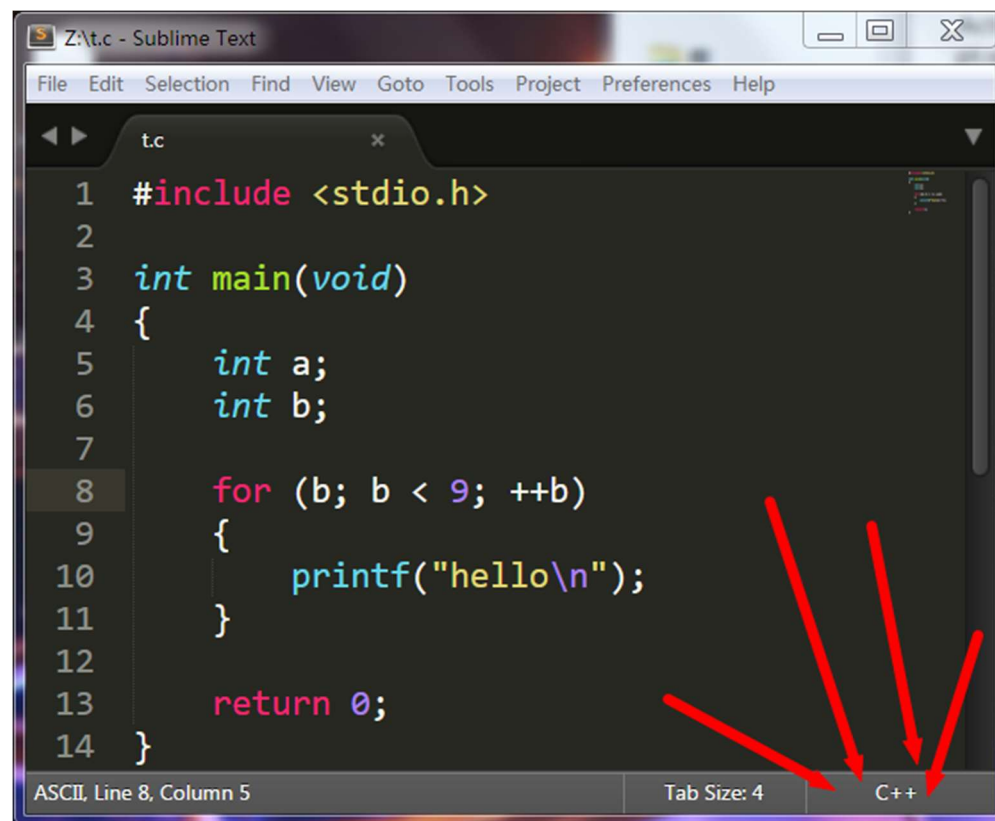
这时候，可以在 Tools -> Build System 下看到刚才新建的 c 了



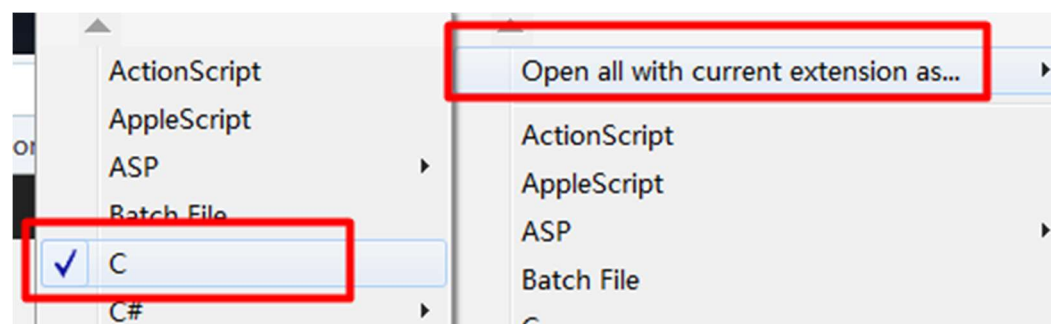
选中后就可以使用了。

Build System 中除了选择具体的编译系统，还可以选择第一个：Automatic 自动选择，会根据打开的文件后缀自动选择。由于默认情况下 .c 文件 sublime 识别为 c++ 类型，所以使用自动选择的时候还需要修改一点：

先用 sublime 打开 .c 文件的时候 默认是 c++ 格式。（注：最新的 3013 版本已经默认是 c 格式，则不必修改）

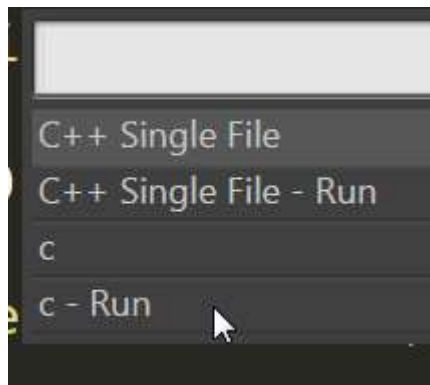


点击红色箭头处的 c++ 选择 Open all with current extension as .. 然后选择 C



这样以后打开 .c 文件就默认是 c 类型

这时候按 Ctrl+Shift+B



第三个 c 就是对应执行配置文件中的第三行 `gcc -Wall $file_name -o $file_base_name` 作用是编译。

第四个 c-Run 对应后面的命令 `gcc -Wall $file -o $file_base_name && start cmd /c \"%{file_path}/{file_base_name} & pause\"`，作用是是在新的 cmd 窗口运行。这样就可以对 scanf 等函数进行输入了。

C++

gcc 虽然可以编译 c++ 代码，但是不能进行 c++ 的连接函数库操作。所以针对 c++ 代码一般使用 g++ 来编译。

方法和上面的 c 语言的配置一样，只要把配置文件中的 gcc 改为 g++，source.c 改为 source.c++，保存文件名 c.sublime-build 改为 c++.sublime-build 就可以了。

这里增加了 `-std=c++11` 选项，是按照 C++11 标准进行编译，不需要的可以去掉，配置文件如下：

```
1      {
2      "encoding": "utf-8",
3      "working_dir": "$file_path",
4      "shell_cmd": "g++ -Wall -std=c++11 \"%file_name\" -o
5      \"%file_base_name\"",
6      "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)??:? (.*)$",
7      "selector": "source.c++",
8      "variants":
9      [
10     {
11     "name": "Run",
12     }
13     ]
14
15
```

```

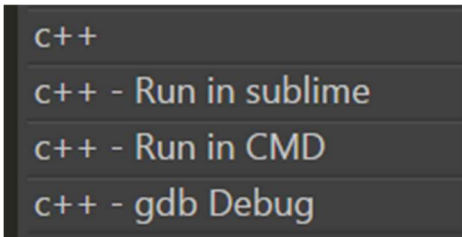
        "shell_cmd": "g++ -Wall -std=c++11 \"$file\" -o
        \"$file_base_name\" && start cmd /c
        \"${file_path}/${file_base_name} & pause\"
    }
}
}

```

实际上，我们可以利用 Variants，来配置多个不同的编译命令。

下面的配置文件有编译，编译并在 sublime 中运行，编译并 cmd 运行和 gdb 调试四个命令。

注：**gdb 调试是打开命令行的方式，这里不支持带空格的源代码文件名和路径**，gdb 的使用可以参考：[gdb 调试新手入门（一） | 雅乐网](#)。要想通过 sublimeGDB 插件实现图形化调试，可以参考 [Sublime Text 3 使用 SublimeGDB 图形化调试 c/c++ 程序](#)。



```

1      {
2      "encoding": "utf-8",
3      "working_dir": "$file_path",
4      "shell_cmd": "g++ -Wall -std=c++11 \"$file_name\"-o
5      \"$file_base_name\"",
6      "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)??:?(.*)$",
7      "selector": "source.c++",
8      "variants":
9      [
10     {
11         "name": "Run in sublime",
12         "shell_cmd": "g++ -Wall -std=c++11 \"$file_name\"-o
13         \"$file_base_name\"&& cmd /c
14         \"${file_path}/${file_base_name}\"
15     },
16     {
17         "name": "CMD Run",
18         "shell_cmd": "g++ -Wall -std=c++11 \"$file\"-o
19         \"$file_base_name\"&& start cmd /c
20         \"\"${file_path}/${file_base_name}\" & pause\"
21     },

```

```

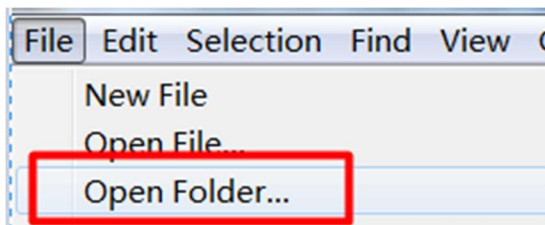
22     {
23     "name": "gdb Debug",
        "shell_cmd": "g++ -g -std=c++11 \"$file\"-o
        \"$file_base_name\"&& start cmd /c gdb
        ${file_path}/${file_base_name} & pause"
    }
  ]
}

```

使用 makefile 编译多个文件

sublime 可以使用 makefile 来编译多个文件，以便支持稍大一点的工程项目。（windows 下面和 linux 下面并不相同，本文介绍适用于 windows）

这个功能只打开单个文件是没有的，只有**打开整个文件夹**



侧边栏中可以看到**打开的文件夹**，确保文件夹中包含 makefile 文件。此时按下 Ctrl+Shift+B，会有 make 的选项。

这里 make 选项执行的是 make，但是 windows 中是没有 make 这个命令的。MinGW\bin 里面的名字是 mingw32-make.exe。解决办法是修改这个文件名改为 make.exe，或者自己新建一个 makefile 的 build 文件。

makefile 的默认编译配置文件在“C:\Program Files\Sublime Text 3\Packages\Makefile.sublime-package”，解压后的 Make.sublime-build 文件中。

我们新建一个编译系统，tool → Build System → New Build System，内容为

```

1     {
2     "shell_cmd": "mingw32-make",
3     "file_regex": "^(..[^:\n]*):([0-9]+):?([0-9]+)?(?:.*)$",
4     "working_dir": "${folder:${project_path:${file_path}}}",
5     "selector": "source.makefile",
6     "syntax": "Packages/Makefile/Make Output.sublime-syntax",
7     "keyfiles": ["Makefile", "makefile"],
8

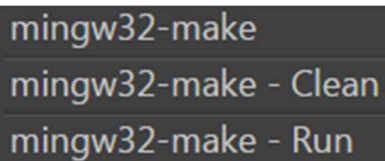
```



```
9      "variants":
10      [
11      {
12      "name": "Clean",
13      "shell_cmd": "mingw32-make clean"
14      },
15      {
16      "name": "Run",
17      "shell_cmd": "mingw32-make run"
18      },
19      ]
20      }
```

保存为 mingw32-make.sublime-build，保存位置和上面 c++ 配置文件位置相同就可以了。

然后打开文件夹后，如果里面有 Makefile 或 makefile 文件，就会有对应的命令



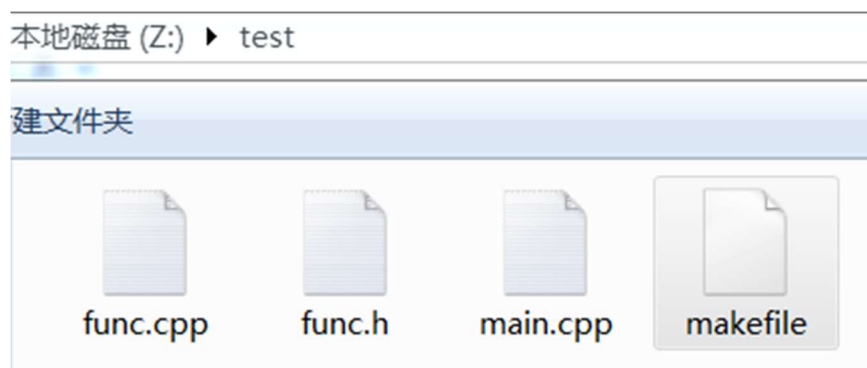
```
mingw32-make
mingw32-make - Clean
mingw32-make - Run
```

由于 windows 下面常用命令和 linux 不同，makefile 也要做对应的修改：

makefile 示例

下面是一个简单的示例：

现在有一个 test 文件夹，里面有 4 个文件



其中 main.cpp 用到了 func.cpp 中的函数。具体代码如下：

main.cpp

```
1      #include "func.h"
```

```
2
3     int main()
4     {
5     output();
6
7     return 0;
8     }
```

func.h

func.cpp

```
1     #include <stdio>
2     void output()
3     {
4     printf("hello world\n");
5     }
```

makefile

```
1     main : main.cpp func.h func.cpp
2
3     clean:
4     del main.exe *.o
5
6     run:
7     mingw32-make && start cmd /c "main.exe & pause"
8
```

注意：makefile 里面的 clean 和 run，和我们自己的配置文件里的“ variants” 下面的命令 mingw32-make clean 和 mingw32-make run 对应的。

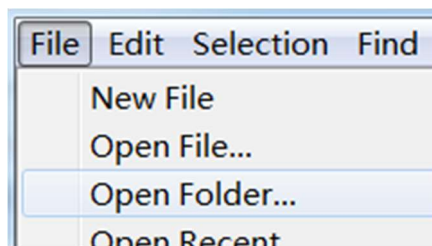
clean 下面使用命令为 del 而不是 linux 下面的 rm

另外，由于 windows 下没有 cc 命令，这里不能出现全部依赖.o 文件的目标。
例如不能有这样的规则：

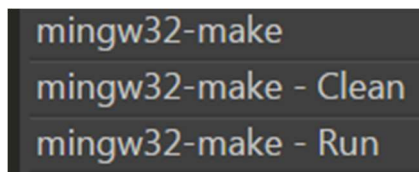
```
1     main : main.o func.o
```

使用这样的规则时，会调用 cc，但是 windows 下面 cc 没法使用，就会报错。

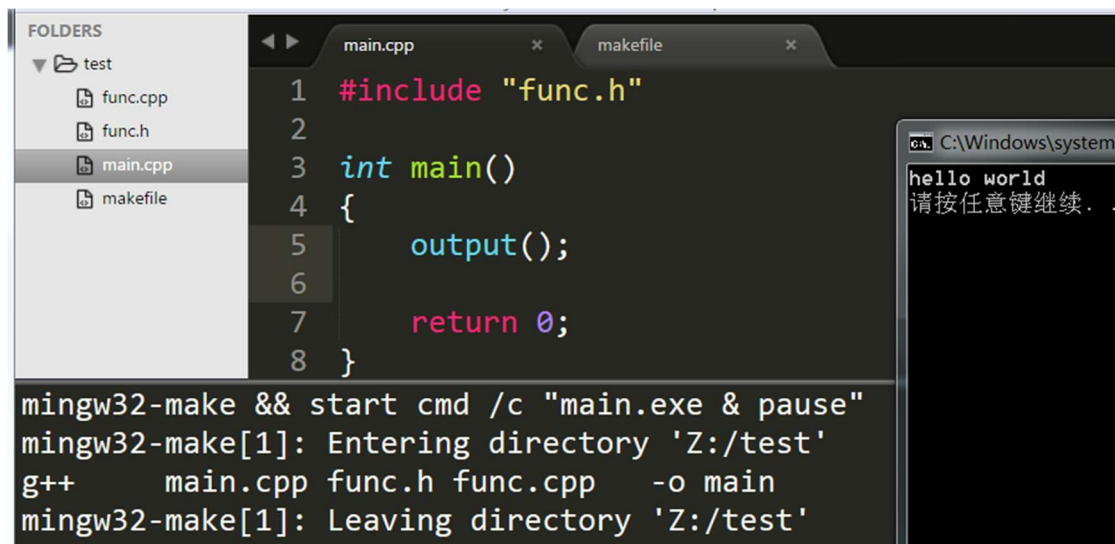
在 sublime text 3 中打开该文件夹（**注意必须打开文件夹，只打开文件没有 make 选项**）



之后使用 Ctrl+shift+B 选择对应的 mingw32-make 命令即可。



Run 对应了编译并运行。



中文编码乱码的问题

感谢 RGB0x000000 同学指出关于中文编码的问题。由于 Sublime Text 3 中文件默认编码格式是 utf-8，而 windows 中的命令行默认编码格式是 GBK。所以代码中出现中文时运行会乱码。

解决思路也很简单，就是让他们编码一致就可以了。

1. 修改cmd 编码为 utf-8

使用 chcp 命令可以查看当前字符集，默认是 936，可以使用 chcp 65001 修改字符集为 utf-8

然而似乎只对当前打开的窗口有效，一个麻烦的办法是每次代码里运行 `system` 来切换字符集（噗）

2. 修改源代码格式为 GBK

Sublime 原生并不支持 GBK 编码，但如果安装了 `ConvertToUTF8` 插件，就可以正确显示 ANSI 或者 GBK 编码的文件。因此，装插件后打开 GBK 编码的源代码文件，也不会乱码。

一个更巧妙地办法是使用编译器的选项 **-fexec-charset** 来设置代码中字符串的编码，这样源文件可以使用 utf-8 编码，只是编译的时候用指定的编码来编译源代码中的字符串。

在编译命令 `gcc` 中加入选项 **-fexec-charset=GBK** 来说明将代码中的字符串按照 GBK 编码，从而和 CMD 窗口一致，也不会乱码。

修改后的 c 语言的配置文件如下：

```
1      {
2      "working_dir": "$file_path",
3      "cmd": "gcc -Wall -fexec-charset=GBK \"$file_name\"-o
4      \"$file_base_name\"",
5      "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)?(?:.*)$",
6      "selector": "source.c",
7      "variants":
8      [
9      {
10     "name": "Run",
11     "shell_cmd": "gcc -Wall -fexec-charset=GBK \"$file\"-o
12     \"$file_base_name\"&& start cmd /c
13     \"$file_path\"/\"$file_base_name\" & pause\"
14     }
15     ]
16     }
```

但是加入这个选项后，如果要编译的不是 utf-8，而是 GBK，必须还要加入 `-finput-charset=GBK` 选项来制定源代码的编码格式，否则会提示错误

error: converting to execution character set: Illegal byte sequence.

而加入这个选项后编译 utf-8 又会乱码