本笔记是以C++为主，用来练习makefile的。使用VSCode + g++编译器

# 示例流程预览

新建一个文件夹，使用vscode打开，新建一个main.cpp和一个Makefile(M大写)，在main.cpp中输入一个打印程序，建立一个基本的CPP程序框架

电脑萤幕画面

描述已自动生成

在makefile中输入

图形用户界面, 文本

描述已自动生成

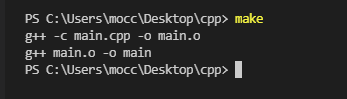
生成的main 是通过main.o生成的，使用的指令是g++ main.o -o main

使用g++ 通过main.o 链接生成main

第三行main.o 通过main.cpp编译而来，使用指令g++ -c main.cpp -o main.o

使用g++ 通过main.cpp文件编译生成 main.o文件

在终端输入make



可以看到左侧资源管理器中出现了main.o和main.exe

图形用户界面, 文本, 应用程序

描述已自动生成

在终端中输入./main.exe即可运行main.exe

图形用户界面, 文本

描述已自动生成

可以看到输出的结果和程序预想的一样

# 多文件编译

文本

中度可信度描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成屏幕的截图

描述已自动生成图形用户界面, 文本, 应用程序

描述已自动生成

# Makefile规则

gcc [选项] [文件名字]

主要选项如下：

-c：只编译不链接为可执行文件，编译器将输入的.c 文件编译为.o 的目标文件。

-o：<输出文件名> 用来指定编译结束以后的输出文件名，如果使用这个选项的话 GCC 默

认编译出来的可执行文件名字为 a.out。

-g：添加调试信息，如果要使用调试工具(如 GDB)的话就必须加入此选项，此选项指示编

译的时候生成调试所需的符号信息。

-O：对程序进行优化编译，如果使用此选项的话整个源代码在编译、链接的的时候都会进

行优化，这样产生的可执行文件执行效率就高。

-O2：比-O 更幅度更大的优化，生成的可执行效率更高，但是整个编译过程会很慢。

本质是每遇到一个tab则创建一个shell脚本去运行它

# 变量赋值等号=

name=qwe

creename=$(name)

name=asd

最后creename输出的值是asd

等号赋值的意思大概就是可以后期更改，使用最近一次更改的值

# 变量赋值等号:=

name=qwe

creename:=$(name)

name=asd

print:

@echo “creename=$(creename)”

加入@之后可以不显示echo这一行，直接输出creename=值

最后creename输出的值是qwe

冒号等号赋值表示只使用前面设置的，不管如何变都不会更改

# 变量赋值等号?=

curname?=qwe

如果curname前面没有被赋值，那么这个变量的值就是qwe，如果前面赋值了，则curname的值就是之前的值

# 变量赋值等号+=

Object=main.o input.o

Object+=calue.o

+=相当于追加字符串

# 自动化变量$@ $% $< $? $^ $+ $#

# 规整输出文件的目录

因为没有更改输出文件位置，所以编译生成的文件全部放在主目录，不方便管理

所以我们需要指定输出文件的目录