## g++ 和 gcc

gcc主要用于编译c相关的文件

g++ 主要用于编译c++相关的文件

* -c代表只是编译，c是compile的缩写

例如`gcc xx.cpp -c`

xx.cpp 转换成目标文件 xx.o，这个目标文件包含了编译后的机器代码，但还不能独立执行。

* **-o 选项**: -o 是 "output" 的缩写，它告诉编译器将生成的文件（通常是可执行文件或目标文件）的名称指定为紧随 -o 选项之后的名称。

`g++ main.cpp -o my\_program`

会将 main.cpp 编译并链接成一个名为 my\_program 的可执行文件。如果不使用 -o 选项，默认情况下编译器会将生成可执行文件命名为 a.out（windows为a.exe）。

* g++ xx.cpp 或者gcc xx.c

其实就是先执行了编译生成“xx.o”文件，然后链接生成可执行文件。

【注意】在c++/c中，打包为可执行文件的时候，只允许有一个main函数。

## Makefile

由于手动在命令行一个个写非常麻烦，所以就产生了Makefile

分为“目标”+“依赖”

格式如下，注意g++是用TAB作为首行：

**Version1:**

# version1  
main: main.cpp test01.cpp

g++ -o main main.cpp test01.cpp

然后到对应的目录，执行make，即生成可执行文件main.exe（linux是二进制）

修改某个.cpp文件，再执行命令make，即可刷新可执行文件hello

缺点：文件多时，编译时间长

**Vrsion 2**

# VERSION 2

# CXX是C++ Compiler的缩写。

CXX = g++ # 指定编译器类型

TARGET = hello

OBJ = main.o printhello.o factorial.o

# make时执行g++ 先找TARGET，TARGET不存在找OBJ，OBJ不存在，编译三个.cpp文件生成.o文件

# 然后再编译OBJ文件，生成可执行文件hello，TARGET依赖OBJ，以此类推

$(TARGET): $(OBJ)

$(CXX) -o $(TARGET) $(OBJ)

# main.o这样来的，编译main.cpp生成

main.o: main.cpp

$(CXX) -c main.cpp

printhello.o: printhello.cpp

$(CXX) -c printhello.cpp

factorial.o: factorial.cpp

$(CXX) -c factorial.cpp

**version 3**

# VERSION 3

CXX = g++

TARGET = hello

OBJ = main.o printhello.o factorial.o

# 编译选项，显示所有的warning；CXXLAGS是C++ Compiler Flags的缩写。

CXXLAGS = -c -Wall

# **$@表示是冒号前面的TARGET，$^表示的是冒号后OBJ的全部.o依赖文件**

$(TARGET): $(OBJ)

$(CXX) -o $@ $^

# $<表示指向%.cpp依赖的第一个，但是这里依赖只有一个

# $@表示指向%.o

%.o: %.cpp

$(CXX) $(CXXLAGS) $< -o $@

# 为了防止文件夹中存在一个文件叫clean

.PHONY: clean

# -f表示强制删除，此处表示删除所有的.o文件和TARGET文件

clean:

rm -f \*.o $(TARGET)

**version 4**

# VERSION 4

CXX = g++

TARGET = hello

# 所有当前目录的.cpp文件都放在SRC里面

SRC = $(wildcard \*.cpp)

# 把SRC里面的.cpp文件替换为.o文件

OBJ = $(patsubst %.cpp, %.o,$(SRC))

# -Wall 是warning all 显示所有警告信息

CXXLAGS = -c -Wall

$(TARGET): $(OBJ)

$(CXX) -o $@ $^

%.o: %.cpp

$(CXX) $(CXXLAGS) $< -o $@

# .PHONY 伪目标，防止有一个同名的clean文件

.PHONY: clean

clean:

rm -f \*.o $(TARGET)

**常用选项**

* **-I**：指定头文件目录路径。
* **-L**：指定库文件目录路径。
* **-l**：指定要链接的库，如-lmylib。