关于makefile

makefile关系到了整个工程的编译规则。一个工程中的源文件不计数，其按类型、功能、模块分别放在若干个目录中，makefile定义了一系列的规则来指定，哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作，因为makefile就像一个Shell脚本一样，其中也可以执行操作系统的命令。makefile带来的好处就是——“自动化编译”，一旦写好，只需要一个make命令，整个工程完全自动编译，极大的提高了软件开发的效率。

makefile规则

```
target ... : prerequisites ...
             command
             ...
             ...
```

target也就是一个目标文件，可以是Object File，也可以是执行文件。还可以是一个标签（Label），对于标签这种特性，在后续的“伪目标”章节中会有叙述。

- prerequisites就是，要生成那个target所需要的文件或是目标。
- command也就是make需要执行的命令。（任意的Shell命令）

这是一个文件的依赖关系，也就是说，target这一个或多个的目标文件依赖于prerequisites中的文件，其生成规则定义在command中。说白一点就是说，prerequisites中如果有一个以上的文件比target文件要新的话，command所定义的命令就会被执行。这就是Makefile的规则。也就是Makefile中最核心的内容。

示例

```
//main.cpp
```

```
#include "test.h"
int main(){
 fun();
 return 0;
}
```

```
//test.h
#ifndef TEST_H_
#define TEST_H_
#include <stdio.h>
void fun();
#endif
```

```
//test.cpp
#include "test.h"
void fun(){
 printf("hello world\n");
}
```

```
//makefile
objects = main.o test.o
#依赖关系
test:$(objects)
#执行的shell命令
 g++ -o test $(objects)
#注意用tab键
#依赖关系
main.o:test.h

#清除生成的.o文件和test可执行文件
.PHONY : clean
clean :
 -rm test $(objects)
#注意用tab键
```

- 在该文件夹目录下，运行make
  生成可执行问价以及中间的.o文件

- ./test
  运行可执行文件
- make clean
  可删除编译生成的文件

参考博客

https://blog.csdn.net/liang13664759/article/details/1771246（写的很详细）