1. Makefile

（一）概述

一个工程中的源文件不计其数，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](https://baike.baidu.com/item/Shell%E8%84%9A%E6%9C%AC)一样，也可以执行操作系统的[命令](https://baike.baidu.com/item/%E5%91%BD%E4%BB%A4/8135974)。

（二）Makefile文件的优势

为工程编写Makefile 的好处是能够使用一行命令来完成“自动化编译”，一旦提供一个（通常对于一个工程来说会是多个）正确的 Makefile。编译整个工程你所要做的事就是在shell 提示符下输入make命令。整个工程完全自动编译，极大提高了效率。

（三）Makefile文件写法

1. 命名规则

Makefile或者是Makefile都可以。

2. makefile的三要素

（1）目标

（2）依赖

（3）规则命令

3. makefile文件初始写法

目标：依赖

tab键 规则命令

目标是我们要生成或者操作的文件，依赖是我们需要的文件，规则命令是我们要给出的操作命令。

如：

qpp:main.c

gcc -o app -I./include main.c

这种写法过于简单，且在某一个依赖文件修改时，所有的依赖文件都要重新编译，非常不方便。

4. makefile升级写法

（1）考虑将编译过程分解，先生成.o二进制文件，再生成目标文件。当某一个.o二进制文件更新时，只需要重新编译一个文件和目标文件，其他文件没有更改。

（2）规则命令是递推的，即如果目标通过命令规则找不到相应的依赖文件，就要找是否有生成该依赖文件的命令规则，直到找到需要的文件为止。

（3）依赖文件如果比目标文件的日期新，则重新生成目标文件。

（4）若依赖文件过多，可以用变量来替换，在使用是用$(obj)的方式使用

（5）makefile默认处理第一个目标

如： obj = main.o add.o

app:$(obj)

gcc -o app -I ./include main.o add.o

makefile文件如下所示：

obj = main.o

qpp:$(obj)

gcc -o app -I./include main.o

main.o:main.c

gcc -c main.c -I ./include

可以看到，目标文件app依赖于main.o，main.o依赖于main.c。

5. makefile的函数

（1）wildcard

进行文件的匹配。如，我们想查找所有的c文件，则可以用wildcard匹配所有的c文件。

srcFiles=$(wildcard \*.c)

使用的时候，用$(srcFiles)即可。

（2）patsubst

进行内容的替换，如将所有的c文件用对应的o文件替换。

如：

srcFiles = $(wildcard \*.c)

objFiles = $(patsubst %.c, %.o, $(srcFiles))

使用的时候，依然是 $(objFiles)即可。

可以看出，式子中srcFiles找出了所有的c文件，objFiles将所有的c文件替换为了o文件。

%为通配符，表示相同的内容。

6. makefile的变量

（1）$@

代表目标

（2）$^

代表全部依赖

（3）$<

代表第一个依赖

（4）$?

代表第一个变化的依赖

变量只能在规则中使用，因为对应的是目标和依赖。

使用：

如：

main.o:main.c

gcc -c main.c -I ./include

可以转换为：

%.o:%.c

gcc -c $< -I ./include

7. makefile文件再升级

之前写的makefile文件通过函数和变量可以再升级。

srcFiles=$(wildcard \*.c)

objFiles=$(patsubst %.c, %.o, $(srcFiles))

app:$(objFiles)

gcc -o app -I ./include $(objFiles)

%.o:%.c

gcc -c $< -I ./include -o $@

整体变得十分简洁高效。

8. 注意事项

（1）规则前加@表示不显示该命令

（2）规则前加“-”表示如果该规则报错，仍然继续执行

（3）如果makefile中的目标和目录中的某个文件名称相同，且该目标只是为了执行命令，而不是生成对应的目标文件。我们需要为了防止歧义，在makefile的该目标上，标注 .PHONY，表示该目标为伪目标。

如下所示：

如目标clean和目录中的clean文件有冲突，则在makefile中声明

.PHONY:clean

（4）make指令执行makefile文件中的规则指令，默认第一个目标，如要指定目标，则在make时需指定该目标名称

（5）目录下有多个makefile文件时，我们可以用make -f makefileName指定执行哪个makefile文件。