GNU make工作时的执行步骤如下：

1. 读入所有的Makefile。
2. 读入被include的其它Makefile。
3. 初始化文件中的变量。
4. 推导隐晦规则，并分析所有规则。
5. 为所有的目标文件创建依赖关系链。
6. 根据依赖关系，决定哪些目标要重新生成。
7. 执行生成命令。

1-5步为第一个阶段，6-7为第二个阶段。第一个阶段中，如果定义的变量被使用了，那么，make会把其展开在使用的位置。但make并不会完全马上展开，make使用的是拖延战术，如果变量出现在依赖关系的规则中，那么仅当这条依赖被决定要使用了，变量才会在其内部展开。下面对makefile的相关问题进行简单介绍。

Makefile的基本结构

Makefile的一般结构：

target…….. : dependency………….

command…………………

结构中各部分的含义：

1. target(目标): 一个目标文件，可以是Object文件，也可以是可执行文件。还可以是一个标签(Label)。
2. dependency (依赖)：要生成目标文件(target)所依赖哪些文件
3. command (命令)： 创建项目时需要运行的shell命令（注意：命令(command)部分的每行的缩进必须要使用Tab而不能使用多个空格

Makefile实际上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dependency中的文件，其生成规则定义在命令command中。如果依赖文件(dependency)中有一个以上的文件比目标(target)文件要新的话，shell命令(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。

例如， 假设有一个C源文件test.c，该源文件包含有自定义的头文件test.h, 则目标文件test.o明确依赖于两个源文件： test.c 和 test.h。 如果只希望利用gcc命令来生成test.o目标文件，这时，就可以利用如下的makefile来定义test.o的创建规则：

#This makefile just is a example.

test.o: test.c test.h

gcc -c test.c

从上面的例子注意到，第一个字符为#的行表示注释行。第一个非注释行指定test.o为目标，并且依赖于test.c和test.h文件。随后的行指定了如何从目标所依赖的文件建立目标。

当test.c或test.h文件在编译之后又被修改，则make工具可自动重新编译test.o，如果在前后两次编译之间，test.c和test.h均没有被修改，而且test.o还存在的话，就没有必要重新编译。这中依赖关系在多源文件的程序编译中尤其重要。通过这种依赖关系的定义，make工具可避免许多不必要的编译工作。

一个makefile文件中可定义多个目标，利用make target命令可指定要编译的目标，如果不指定目标，则使用第一个目标。通常，makefile中定义有clean目标，可用来清除编译过程中的中间文件。

#This makefile just is a example.

test.o: test.c test.h

gcc -c test.c

clean:

rm -f \*.o

运行make clean时，执行rm -f \*.o命令，删除编译过程中生成的所有中间文件。

Makefile的基本内容：

Makefile一般包含：显式规则、隐晦规则、变量定义、文件指示和注释等第五个内容。

1. 显式规则： 显式规则说明如何生成一个或多个的目标文件。这是由Makefile的书写者明显指出，要生成的文件，文件的依赖文件，生成的命令。
2. 变量定义：在Makefile中可以定义一系列的变量，变量一般都是字符窜，当Makefile被执行时，变量的值会被扩展到相应的引用位置上。
3. 隐晦规则：由于GNU make具有自动推导功能，所以隐晦规则可以比较粗糙地简略地书写Makefile，然后由GNU make的自动推导功能完成隐晦规则的内容。
4. 文件指示。其包括了三个部分，一个是在一个Makefile中引用另一个Makefile,就像C语言中的include一样，另一个是指根据某些情况制定Makefile中有效部分，就像C语言中的预编译#if一样；还有就是定义一个多行的命令。
5. 注释：Makefile中只有行注释，和UNIX的shell脚本一样，其注释是用”#”字符，如果你要在你的Makefile中使用”#”字符，可以用反斜框进行转义，如： “\#”

Makefile中的变量

1、Makefile中定义的变量，与C/C++语言中的宏一样，代表一个文本字串，在Makefile被执行时候变量会自动地展开在所使用的地方。Makefile中变量可以使用在”目标”, “依赖目标”,

“命令”或者Makefile的其他部分中。

2、Makefile中变量的命名可以包含字符、数字、下划线(可以是数字开头)，但不应该含有”:”、

“#”、”=”或是空字符(空格、回车等)。

1. Makefile中变量是大小写敏感的，传统的Makefile的变量名是全大写的命令方式。
2. 变量在声明时需要给予初值，而在使用时，需要在变量名前加上”$”符号

#makefile test for hello program.

#written by Emdoor

CC=gcc

CFLAGS=

OBJS=hello.o

all: hello

hello: $(OBJS)

$(CC) $(CFLAGS) $(OBJS) -o hello

hello.o: hello.c

$(CC) $(CFLAGS) -c hello.c -o $(OBJS)

clean:

rm -rf hello \*.o

GNU make的主要预定义变量

GNU make 有许多预定义的变量，这些变量具有特殊的含义，可在规则中使用。以下给出了一些主要的预定义变量，除这些变量外，GNU make还将所有的环境变量作为自己的预定义变量。

$@ ------表示规则中的目标文件集。在模式规则中，如果有多个目标，那么，”$@”就是匹配于目标中模式定义的集合。

$% -------仅当目标是函数库文件中，表示规则中的目标成员名。例如，如果一个目标是”foo.a(bar.o)”，那么，”$%”就是”bar.o”, “$@”就是”foo.a”。如果目标不是函数库文件(Unix下是[.a]，windows下式[.lib])，那么其值为空。

$< ------依赖目标中的第一个目标名字。如果依赖目标是以模式(即”%”)定义的，那么

“$<”将是符合模式的一系列的文件集。注意，其是一个一个取出来的。

$? --------所有比目标新的依赖目标的集合。以空格分隔

$^ ---------所有的依赖目标的集合。以空格分隔。如果在依赖目标中有多个重复的，那么

这个变量会去除重复的依赖目标，只保留一份。

$+ ----------这个变量很像”$^”，也是所有依赖目标的集合。只是它不去重复的依赖目标。

命令的变量

1. AR 函数库打包程序。默认命令是 “ar”。
2. AS 汇编语言编译程序。默认命令是”as”。
3. CC C语言编译程序。默认命令是”cc”。
4. CXX C++语言编译程序。默认命令是”g++”。
5. CO 从RCS文件中扩展文件程序。默认命令是”co”。
6. CPP C程序的预处理器（输出是标准输出设备）。默认命令是”$(CC) -E”。
7. FC Fortran和Ratfor的编译器和预处理程序。默认命令是”f77”。
8. GET 从SCCS文件中扩展文件的程序。默认命令是”lex”。
9. LEX Lex方法分析器程序(针对于C或Ratfor)。默认命令是”f77”。
10. PC Pascal语言编译程序。 默认命令是 “pc”。
11. YACC Yacc文法分析器（针对于C程序）。默认命令是”yacc”。
12. YACCR YACC文法分析器（针对于Ratfor程序）。默认命令是”yacc -r”。
13. MAKEINFO 转换Textinfo源文件(.texi)到Info文件程序。默认命令是”make info”。
14. TEX 从TeX源文件创建TeX DVI文件的程序。默认命令是 “tex”。

15、TEXI2DVI 从Texinfo源文件创建TeX DVI文件的程序。默认命令是”texi2dvi”。

16、WEAVE 转换Web到TeX的程序。默认命令是”weave”。

17、CWEAVE 转换C Web到TeX的程序。默认命令是”cweave”。

18、TANGLE 转换C Web到C。默认命令是”ctangle”。

19、CTANGLE 转换C Web到C。默认命令是 “ctangle”。

20、RM 删除文件命令。默认命令是”rm -f”。

命令参数变量：

下面的这些变量都是相关上面的命令的参数。如果没有指明其默认值，那么其默认值都是空。

ARFLAGS 函数库打包程序AR命令的参数。默认值是“rv”。

ASFLAGS 汇编语言编译器参数。（当明显地调用“.s”或“.S”文件时）。

CFLAGS C语言编译器参数。

CXXFLAGS C++语言编译器参数。

COFLAGS RCS命令参数。

CPPFLAGS C预处理器参数。（ C 和 Fortran 编译器也会用到）。

FFLAGS Fortran语言编译器参数。

GFLAGS SCCS “get”程序参数。

LDFLAGS 链接器参数。（如：“ld”）

LFLAGS Lex文法分析器参数。

PFLAGS Pascal语言编译器参数。

RFLAGS Ratfor 程序的Fortran 编译器参数。

YFLAGS Yacc文法分析器参数。

隐含规则