相关makefile资料可以参考《跟我一起写makefile》，这篇文档只是把自己容易忽视和犯错误的地方重新整理一下，并按照自己的理解描述。 学完makefile之后参考韦东山的课程自己写了一个通用的makefile，并且有较为详细的注释，很多项目中是可以直接拿来使用的，相关资料在makefile实验/ 目录下。

* GNU 的 make 工作时的执行步骤入下：（想来其它的 make 也是类似）  
  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。

这里需要注意的是执行生成命令的时候，所有的变量值已经被确定了

* 通配符的使用： \* 和 % , 这两个很容易搞错

\*.c 指的是 所有的 .c 结尾的文件，这会是一堆文件， 一般都是搭配 rm -f \*.c

%.c 指的是匹配任意字符的.c 文件， makefile中 可以这样使用

%.o: %.c

$(CC) $(CFLAGS) -Wp,-MD,.$@.d -c -o $@ $<

这个对于main.o 依赖的就是main.c ， % 就相当于一个替换关系：% 匹配main ，该规则中所有% 都用main 来替换

* 自动化变量 $@ $< $^

$@ 表示目标

$< 表示第1个依赖文件

$^ 表示所有依赖文件

实际使用可以参考 makefile实验/ makefile.build

* 几种变量的赋值

:= # 即时赋值

= # 延时赋值

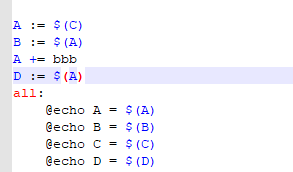
?= # 延时变量, 如果是第1次定义才起效, 如果在前面该变量已定义则忽略这句

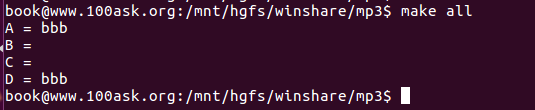
+= # 附加, 它是即时变量还是延时变量取决于前面的定义

对于 := 和 = 其实操作符右边如果不是变量的话都是可以一样的效果，也就是这条语句就结束后 操作符左侧的变量值已经定下来了。

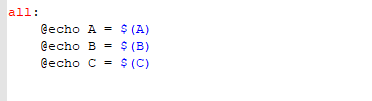
唯一不同的地方在于操作符右侧是变量！

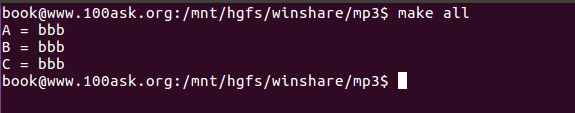
：= 如果操作符右侧是变量，例如 A := $(B) 在make刚开始工作第3步的时候扫描变量的时候变量A的值 会立即确认下来，也就是如果B 截至到这里没有被赋值 ，那么此时 A就是空。





= 如果右边操作符是变量的话， make扫描变量的时候，到这个地方的时候就暂时跳过，等到makefile扫描变量结束后，再确定这个变量的值。





+= 这个变量之前赋值方式是有关的。

例如 ： A := aaa

A += bbb 这个就相当于 A := $(A) bbb

A = aaa

A += bbb 此时相当于 A = $(A) bbb （这个看上去会出现递归调用错误，但实际上makefile 帮我们处理掉这个错误，也就是相当于变量第一次扫描结束后在增添上+= 的部分）

* .PHONY

用来指出是伪目标，也就是说指定本来不是一个实际文件，而只是一个标签。用.PHONY指定之后不管这个文件是否存在，都要重新生成。

其实主要作用就是防止你的伪目标名字和实际文件或者文件夹名字重名。如果不用PHONY指定的话，恰好这个伪目标和文件或者文件名重名就不会再生成了。

这个当时在调试