3.4 MAKEFILE变量

如果定义了MAKEFILES变量，make考虑将值作为名字序列。者与include指令很小，不同的目录会寻找这些文件。除此之外，默认的目标从没有从这些makefile中提。

MAKEFILEs的主要作用是在make的invocation递归中通信。通常，我们不希望从顶层 make invocation设置环境变量，因为最好不要从外面mess makefile。然而，如果你没有特殊的makefile然后运行make，在MAKESFILES中的makefile会比较有用，来帮助内置的隐式规则更好的工作，比如定义搜索路径。

3.5 makefiles是怎样remade的?

一些makefile可以从其他文件中remade，如果一个makefile可以从其他文件中remade，那么你可能需要make的得到最新的版本的makefile

处于这个目的在将所有makefile读入之后，make会考虑这样的母便，并尝试更新。如果makefile中说明了怎样更新，或者有隐式规则，那么就会被更新。在所有的makefile被监察了之后，如果其中的一些被修改了，make从一个干净状态开始，然后重新读取所有的makefile。每个restart 都会是MAKE\_RESTARTS变量更新。

如果你知道，你的makefile中一个或者多个不能被remade，是你不想隐式规则找到他们（可能处于编译效率原因），你可以使用正常的方法来避免隐式规则来被搜索，例如，你可以写一个显式规则，将makefile作为target,然后空recipe。

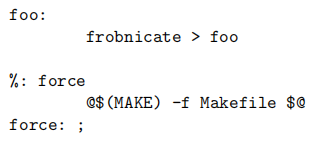
如果makefile声明一个双冒号来根据recipe remake一个文件，这个文件总是需要被remade。在makefiles中，一个有双冒号的makefile在每次make都会被运行。这回导致无限循环，make会不断地remake makefile，并不再做其他事。从而，为了避免这个问题，make不再尝试remake文件。

3.6 另外一个makefile的重写部分

有时候一个makefile与另一个makefile相似是非常有用的，你可以使用include在另一个总包含这个，并增加更多的变量或者定义。然而， 两个target的recipe不同也是可以的

在包含makefile，你可以用一个匹配模型，来remake任何target，make应该。

例如，如果你有一个makefile，说运行foo，你可以写一个makefile包含下面内容



如果你make foo，那么会找到GNUmakefile，读取，运行。如果你make bar，那么没有target，%是一个匹配付，可以匹配到，从。相似的，对于任何其他target，

%规则需要force，为了确保这个recipe会在target存在的情况下运行，我们给force一个空recipe来避免make搜索隐式规则。

3.7 make怎样读取makefile

GUNmake通过两个阶段进行这个工作。在第一个阶段，读取所有的makefile文件，内部化所有变量以及值，隐式规则和显式规则，建立依赖图。第二个阶段，make使用内部数据来决定哪个target需要被更新，并允许合适的recipes来更新。

这两个阶段有一个直接的影响变量和函数扩展。

我们说一个扩展是immediate如果这在第一个阶段发生。Defferred如果其他阶段。Defferred的扩展在第二阶段需要的时候进行。

对于append操作，如果变量之前是一个简单变量（：=，：：=），那么就是immediate，否则是deferred。

对于shell赋值操作，右边被immediate，结果存在右边，变量编程了单个变量

**条件指令**

条件指令immediately 被解析，这意味着，自动变量不能够用作条件指令，因为自动变量在recipe的时候才有用。如果你需要自动变量在条件指令中，你必须把条件放在recipe中并使用shell 语句

**规则定义**

一个规则总是以同样的方式扩展，无论形式

也就是说，targe以及依赖都是immediate，recipei是deferred。

3.8 makefile是怎样解析的

对于某一行

1.读取完整的逻辑行

2.去除注释

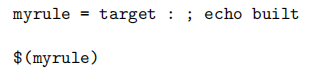
3.如果以recipe prefix字符开始，而且我们在rule环境中，增加line到当前recipe并读取下一行。

4.expand immediate context

5.扫描分割符号行，比如，来决定line是一个宏分配还是一个rule

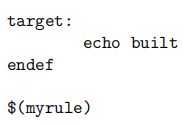
6.内部化结果操作，读取下一行

宏的一个后果是，一个宏可以扩展到整个规则中：



然而，这不会work，因为make并不在expandhi后re-split line。





上面的结果，targe会依赖echo和built，就行而不是一个规则和一个recipe新航会在扩展结束后被鼓励。为了合适的扩展多行宏，你必须使用eval函数.

3.9 二次扩展

直线我们知道，GNU有两个阶段。GNU也能够进行第二次扩展。为了实现第二次扩展需要。。。Pass

4.2 依赖的类型

GNU make能够理解两种依赖类型：正常依赖和order-only 依赖。一个正常依赖有两个个statement，首先，加载一个order，其中recipe会被invoked：所有依赖的recipe会被执行，然后target的recipe会运行；第二，施加一个依赖关系，如果任何依赖比新的target新，那么会被重写加载。

通常，你想要：如果一个依赖被更新，那么target也应该被更新。

有时，你坑你想要不同的情况，其中，你想要增加一个特殊的顺序，从而不强制某一个更新，这种情况下，你需要定义order-only依赖，order-only依赖可以通过管道符号声明。管道的左边是normal，右边是order-only

正常的依赖部分可能是空的。当然，你可能对同样的target声明多行依赖：他们合适的append。依赖被append到normal依赖列表中。Order-only依赖append到order-only依赖中。注意，如果你声明一个文件为normal和order-only，那么就normal 优先。

考虑一个例子，其中你的target在不同的目录中被放置，这个目录在make之前不存在，这种情况下，你想要目录在target产生之前创建。

完成这个的一种方式是，只使用order-only依赖。

4.3 使用wild card 字符

单个文件名字可以声明很多文件，使用wildcard子u。

make支持target和依赖中的wildcard 扩展

4.3.1 wildcard 例子

Wildcard可以用在规则的rule中，他们被shell扩展。

Wildcard也用在依赖中。

这个规则使用print作为空母便文件

Wildcard在边路中没有用，需要显式说明



4.3.1 wildcard的陷阱

现在有一个自然的方法来使用wildcard扩展，结果可能与你期待的不同。假设你依赖是目录中所有文件，你这样写：

Object的值是’\*.o’，可以扩展为所有的.o文件

但是删除的时候会出问题

4.3.3 wildcard函数

Wildcard函数在规则中自动发生，但是wildcard并不在变量中或者函数参数中。

我们可以概率C的list到obj文件list通过代替。

因

4.4 寻找依赖的目录

对于大系统，通常需要不source放置到单个目。Make通过目录搜索功能来寻找依赖。当你在目录中redistribute，你并不需要改变每个规则，只需要改变搜索路径。

4.4.1VPATH 搜索所有的依赖

Vpath变量横眉make搜索的目录。目录包含依赖不再当前目录的依赖文件，然而，make使用，

4.4.2 vpath指令

与vpath

4.4.5 目录搜索以及隐式规则

1. 使用隐式规则

Remake目标文件经常被使用，例如一个方法来make一个对象文件是从一个C文件中，。

隐式规则告诉说明一种习惯性行为，从而你不需要详细的说明他们。例如，C编译有隐式规则，文件名字决定使用哪一个隐式规则。例如，C编译一般将a.c变成a.o，从而make应用隐式规则，当它看到文件名字的这种组合。

隐式规则链可以按照序列实现，例如，make会通过a

内置隐式规则使用多个变量，从而，通过改变这些变量的值，你可以改变隐式规则的工作方式。例如CLFAGS控制给C编译器的

你可以通过书写模板规则来定义自己的隐式规则。

10.1 使用隐式规则

为了运行make寻找一种习惯的方式来

如果一个隐式规则有一个target 模板和依赖模板。那么同样的target 模板可能有很多隐式规则。比如。实际上使用的规则是以来崔泽或者可以被生成的规则。因此，如果有一个文件，

10.5 定义和重定义模板规则

你通过模板规则，定义隐式规则。不过，目标包含%符号。目标被认为是匹配文件名字的模板。%可以匹配任意地非空子自负床。然而其他字符串只匹配他们自己。依赖相似的使用%来显式他们的相关的目标名字。

因此显示了。

注意，扩展在变量和函数扩展之后，

10.5.1 模板规则介绍。一个模板来匹配文件名字。

匹配到的字符串叫做stem。

表示名字。为了使用模板规则。

如果多个匹配到了某个模板，那么会选取最适合的那个。

10.5.4 模板是怎样匹配的

当一个模板的target是%，我们把这个叫做匹

10.5.6 取消隐式规则

你可以通过定义一个新的模板规则。这个模板的target和依赖相同，但是recipe不同。

你可以通过下面形式定义一个没有recipe的rule（如果不写，那么会用隐式规则）

10.6 定义万不得已的默认规则

你可以定义一个没有任何依赖的默认规则。这就像其他的模板规则意义，唯一特殊的是，这会匹配任意的target。因此，这个的规则的recipe用于所有的没有recipe的target和依赖。

例如，当测试makefile，你可能不关心源文件是否有数据，而只是关系。

  
 会让所有被需要配创建的依赖被自动创建。

你可以定义一个recipe，为了target，这个target没有规则，甚至声明reicpe。