模式规则和自动化变量

学习参考<跟我一起写Makefile>.

在使用makefile时，会有一些经常使用且规则都类似的编译命令，比如把一系列.o文件都需要对应的.c文件来进行编译，我们可以定义这样的“隐含模式规则”，使得我们不需要明确指出目标的依赖和执行命令，比如当makefile发现需要用到.o文件而没有找到时，它会自动按照模式规则中的规定的命令，来推导生成这个.o文件需要执行的命令。

在模式规则的定义中，必然需要一些辅助变量，来替代符合模式的目标文件和依赖文件，来说明要对这些文件执行什么样的命令。自动化变量就是完成这个功能的。自动化变量一共有七类，包括`$@`,`$%`,`$<`,`$^`,`$+`,`$*`,`$?`，分别指代符合某些规则的目标文件或依赖文件等，自己用到过的有三个：

- `$@`，指代规则中的目标文件
- `$<`，指代规则中的依赖目标
- `$^`，指代规则中的所有依赖目标。如果依赖目标中有重复的，那么这个变量会去除重复的依赖目标，只保留一份。

`$@`和`$<`在扩展时会遍历符合规则的文件列表，每次只代表一个文件，而`$^`的值是一个文件列表。

下面这个例子，指定了需要用到.o文件时，就去编译对应的.c文件。其中目标中的`%`值决定了依赖中的`%`值。比如此时依赖中需要`shen.o`则对应的.c文件为`shen.c`

```
%.o : %.c
    $(CC) -c $< -o $@
```

下面是一个一个简单的例子，`main1.c`文件和`main2.c`文件调用了`bar.c`文件和`foo.c`文件中的函数，用makefile对他们编译连接。

```
CC = gcc
targets = main1 main2
objects = bar.o foo.o
.PHONY : all
all : $(targets)

main1 : main1.o $(objects)
    $(CC) -o $@ $^
main2 : main2.o $(objects)
    $(CC) -o $@ $^
%.o : %.c
    $(CC) -c $< -o $@
```

执行`make`命令的结果为：

```
E:\DeskTop\模式test>make
gcc -c main1.c -o main1.o
gcc -o main1 main1.o bar.o foo.o
gcc -c main2.c -o main2.o
gcc -o main2 main2.o bar.o foo.o
```

注意隐含模式规则并不是类似宏一样批量定义一系列目标文件及其依赖，它是说明了当make需要一个文件而没有找到生成这个文件的命令和依赖时，以什么的规则来推导出这个文件。若文件夹中存在一个文件符合.c结尾，他不会自动编译成对应的.o文件。比如上面这个例子，若文件夹中还有`jiji.c`或者`lala.c`，他们是不会被编译成`jiji.o`和`lala.o`的。