# 使用条件判断

使用条件判断，可以让make根据运行时的不同情况选择不同的执行分支。条件表达式可以是比较变量的值，或是比较变量和常量的值。

## 示例

下面的例子，判断$(CC)变量是否“gcc”，如果是的话，则使用GNU函数编译目标。

libs\_for\_gcc = -lgnu

normal\_libs =

foo: $(objects)

ifeq ($(CC),gcc)

$(CC) -o foo $(objects) $(libs\_for\_gcc)

else

$(CC) -o foo $(objects) $(normal\_libs)

endif

可见，在上面示例的这个规则中，目标“foo”可以根据变量“$(CC)”值来选取不同的函数库来编译程序。

我们可以从上面的示例中看到三个关键字：ifeq、else和endif。ifeq的意思表示条件语句的开始，并指定一个条件表达式，表达式包含两个参数，以逗号分隔，表达式以圆括号括起。else表示条件表达式为假的情况。endif表示一个条件语句的结束，任何一个条件表达式都应该以 endif结束。

当我们的变量$(CC)值是“gcc”时，目标foo的规则是：

foo: $(objects)

$(CC) -o foo $(objects) $(libs\_for\_gcc)

而当我们的变量$(CC)值不是“gcc”时（比如“cc”），目标foo的规则是：

foo: $(objects)

$(CC) -o foo $(objects) $(normal\_libs)

当然，我们还可以把上面的那个例子写得更简洁一些：

libs\_for\_gcc = -lgnu

normal\_libs =

ifeq ($(CC),gcc)

libs=$(libs\_for\_gcc)

else

libs=$(normal\_libs)

endif

foo: $(objects)

$(CC) -o foo $(objects) $(libs)

## 语法

条件表达式的语法为：

<conditional-directive>

<text-if-true>

endif

以及：

<conditional-directive>

<text-if-true>

else

<text-if-false>

endif

其中<conditional-directive>;表示条件关键字，如“ifeq”。这个关键字有四个。

第一个是我们前面所见过的“ifeq”

ifeq (<arg1>, <arg2>)

ifeq '<arg1>' '<arg2>'

ifeq "<arg1>" "<arg2>"

ifeq "<arg1>" '<arg2>'

ifeq '<arg1>' "<arg2>"

比较参数“arg1”和“arg2”的值是否相同。当然，参数中我们还可以使用make的函数。如：

ifeq ($(strip $(foo)),)

<text-if-empty>

endif

这个示例中使用了“strip”函数，如果这个函数的返回值是空（Empty），那么<text-if-empty>;就生效。

第二个条件关键字是“ifneq”。语法是：

ifneq (<arg1>, <arg2>)

ifneq '<arg1>' '<arg2>'

ifneq "<arg1>" "<arg2>"

ifneq "<arg1>" '<arg2>'

ifneq '<arg1>' "<arg2>"

其比较参数“arg1”和“arg2”的值是否相同，如果不同，则为真。和“ifeq”类似。

第三个条件关键字是“ifdef”。语法是：

ifdef <variable-name>

如果变量<variable-name>的值非空，那到表达式为真。否则，表达式为假。当然，<variable- name>同样可以是一个函数的返回值。注意，ifdef只是测试一个变量是否有值，其并不会把变量扩展到当前位置。还是来看两个例子：

示例一：

bar =

foo = $(bar)

ifdef foo

frobozz = yes

else

frobozz = no

endif

示例二：

foo =

ifdef foo

frobozz = yes

else

frobozz = no

endif

第一个例子中，“$(frobozz)”值是“yes”，第二个则是“no”。

第四个条件关键字是“ifndef”。其语法是：

ifndef <variable-name>

这个我就不多说了，和“ifdef”是相反的意思。

在<conditional-directive>这一行上，多余的空格是被允许的，但是不能以[Tab]键做为开始（不然就被认为是命令）。而注释符“#”同样也是安全的。“else”和“endif”也一样，只要不是以[Tab]键开始就行了。

特别注意的是，make是在读取Makefile时就计算条件表达式的值，并根据条件表达式的值来选择语句，所以，你最好不要把自动化变量（如“$@”等）放入条件表达式中，因为自动化变量是在运行时才有的。

而且，为了避免混乱，make不允许把整个条件语句分成两部分放在不同的文件中。