

## 目 录

1	使用隐含规则	2
2	常用的隐含规则	2
3	隐含规则使用的变量	3
4	模式规则	3
4.1	模式规则的介绍 . . . . .	3
4.2	自动化变量 . . . . .	3
4.3	模式匹配的注意点 . . . . .	4

## 1 使用隐含规则

隐含规则是 make 事先约定好的一些东西。如果我们没有写出某个目标的生成规则，make 将自动推导产生这个目标的规则，此时，我们就相当于使用了隐含规则。例子如下：

```
1  # 未使用隐含规则
2  foo: foo.o bar.o
3      cc -o foo foo.o bar.o $(CFLAGS) $(LDFLAGS)
4  foo.o: foo.c
5      cc -c foo.c $(CFLAGS)
6  bar.o: bar.c
7      cc -c bar.c $(CFLAGS)
8
9  # 使用隐含规则
10 foo: foo.o bar.o
11     cc -o foo foo.o bar.o $(CFLAGS) $(LDFLAGS)
```

## 2 常用的隐含规则

一些常用的隐含规则如下所示：

- 编译 C 程序的隐含规则，<n>.o 的依赖目标默认为 <n>.c，命令如下：

```
1 $(CC) -c $(CPPFLAGS) $(CFLAGS)
```

- 编译 C++ 程序的隐含规则，<n>.o 的依赖目标默认为 <n>.cc，命令如下：

```
1 $(CXX) -c $(CPPFLAGS) $(CFLAGS)
```

- 编译汇编的隐含规则，<n>.o 的依赖目标默认为 <n>.s，默认编译器为 as，命令如下：

```
1 $(AS) $(ASFLAGS)
```

- 汇编预处理的隐含规则，<n>.s 的依赖目标默认为 <n>.S，默认编译器为 cpp，命令如下：

```
1 $(AS) $(ASFLAGS)
```

- 链接 object 文件的隐含规则，<n> 的依赖目标默认为 <n>.o，命令如下：

```
1 $(CC) $(LDFLAGS) <n>.o $(LOADLIBES) $(LDLIBS)
```

## 3 隐含规则使用的变量

隐含规则中的变量一般有两种，一种是与命令相关的，一种是与参数相关的。下面介绍一些常用的变量。

与命令相关的变量如下：

- AS，默认值为 as。
- CC，默认值为 cc。
- CXX，默认值为 g++。
- CPP，默认值为 C 程序的预处理器。
- RM，默认值为 rm。

与参数相关的变量如下：

- CFLAGS，C 语言编译器参数。
- CXXFLAGS，C++ 语言编译器参数。
- CPPFLAGS，C 预处理器参数。

## 4 模式规则

### 4.1 模式规则的介绍

在模式规则中，目标和依赖目标都必须有“%”，目标中的“%”值决定了依赖目标中“%”的值。例子如下：

```
1 %.o: %.c; <command>
```

上述例子中，如果要生成的目标是“a.o b.o”，那么“%c”就是“a.c b.c”。

### 4.2 自动化变量

所有的自动化变量如下所示：

- \$@。表示规则中的目标文件集。
- \$%。如果目标是函数库文件，那么该值是规则中的目标成员名，否则该值为空。

- `$<`。表示依赖目标中的第一个目标，如果依赖目标由% 定义，那么该值是一系列的文件集，并且是一个一个取出来的。
- `$?`。表示所有比目标新的依赖目标的集合。
- `$.^`表示所有依赖目标的集合，不允许存在重复的依赖目标。
- `$+`。表示所有依赖目标的集合，允许存在重复的依赖目标。
- `$*`。表示目标模式中“%”及其之前的部分。如果目标中没有模式的定义，只要是目标文件的后缀 make 可以识别，那么 `$*` 就是除了后缀的那一部分，否则 `$*` 就是空值。

### 4.3 模式匹配的注意点

当一个模式匹配包含有斜杠的文件时，那么在进行模式匹配时，目录部分会首先移开，然后进行匹配，成功后，再把目录加回去。例子如下：

```
1 # 有模式 “e%t”，依赖目标的模式是 “c%r”
2 # 目标文件是 “src/eat”
3 # 那么依赖目标文件是 “src/car”
```