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：

```
# 未使用隐含规则
foo: foo.o bar.o
    cc -o foo foo.o bar.o $(CFLAGS) $(LDFLAGS)
foo.o: foo.c
    cc -c foo.c $(CFLAGS)
bar.o: bar.c
    cc -c bar.c $(CFLAGS)

# 使用隐含规则
foo: foo.o bar.o
    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：

```
    $(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：

```
    $(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：

```
    $(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：

```
    $(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：

```
    $(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。