Windows 下的 Makefile 编写（二）宏和预处理的简单示例

作者：cntrump

在 Makefile 中使用宏和预处理能明显提高工作的效率。

宏

宏的语法为：

```
macroname=string
```

macroname 是字母、数字和下划线 (_) 的组合，最多 1,024 个字符且区分大小写。macroname 可以包含调用的宏。如果 macroname 完全是由调用的宏组成的，则正调用的宏不能为空或未定义。

宏的使用：

定义了一个宏之后就可以使用了。使用的方法很简单，如下所示就是一个简单的调用过程：

```
$(macroname)
```

使用括号将宏名称括起来，再在前面加上 $ 符号就可以了。在实际中的使用：

```
objects=stdafx.obj main.obj
Test.exe:$(objects)
    .......
```

如果 string 的长度太长或者需要分行显示。可以使用 \ 。反斜框后紧跟着回车就表示换行：

```
objects=stdafx.obj \
        main.obj
Test.exe:$(objects)
```

nmake 还内置了用于指定文件名的宏，叫作文件名宏。

文件名宏被预定义为依赖项中指定的文件名（而不是磁盘上的完整文件名指定）。在调用时不需要将这些宏括在括号内；只需按如下方式指定 $。

宏	意义
$@	当前所指定的当前目标的全名（路径、基名称、扩展名）。
$$@	当前所指定的当前目标的全名（路径、基名称、扩展名）。仅在作为依赖项中的依赖项时有效。
$*	当前目标的路径和基名称，没有文件扩展名。
$**	当前目标的所有依赖项。

宏	意义
$?	时间戳比当前目标的时间戳晚的所有依赖项。
$<	时间戳比当前目标的时间戳晚的依赖文件。仅在推理规则的命令中有效。

使用文件名宏对编写 Makefile 是很有帮助的，特别是在文件数量多的时候，可以节省大量时间。例如上面的例子，使用的文件名宏后：

```
objects=stdafx.obj \
        main.obj
Test.exe:$(objects)
    link.exe $**
```

这样只需使用 $** 就替代了 Test.exe 所依赖的所有.obj 文件，相当方便。

生成文件预处理

预处理指令不区分大小写。初始感叹号 (!) 必须出现在行首。感叹号后面可以有零个或多个空格或制表符，用于缩进。

下面是经常会用到的预处理：

!MESSAGE text

用来显示一段文本信息，显示的内容就是 text 所指定的内容。

!INCLUDE [<]filename[>]

作用类似于 C++ 中的 #include ，将 filename 包含进来，如果 filename 里的指令可执行则会先执行其中的指令然后再继续。

!IF constantexpression

如果 constantexpression 计算结果为非零值，则处理 !IF 和下一个 !ELSE 或 !ENDIF 之间的语句。

!IFDEF macroname

如果定义了 macroname，则处理 !IFDEF 和下一个 !ELSE 或 !ENDIF 之间的语句。空宏将被视为尚待定义。

!IFNDEF macroname

如果没有定义 macroname，则处理 !IFNDEF 和下一个 !ELSE 或 !ENDIF 之间的语句。

!ELSE[IF constantexpression | IFDEF macroname | IFNDEF macroname]

如果前面的 !IF、!IFDEF 或 !IFNDEF 语句计算结果为零值，则处理 !ELSE 和下一个 !ENDIF 之间的语句。可选关键字提供了进一步的预处理控制。

!ELSEIF

!ELSE IF 的同义词。

!ELSEIFDEF

!ELSE IFDEF 的同义词。

!ELSEIFNDEF

!ELSE IFNDEF 的同义词。

!ENDIF

标记 !IF、!IFDEF 或 !IFNDEF 块的结尾。同一行上 !ENDIF 后面的所有文本被忽略。

!UNDEF macroname

取消定义 macroname。

预处理数量虽然不少，但是很多都有其同义预处理。只需要记忆其中一个就可以了。

最后用一个简单的示例来说明宏和预处理的应用，附件中的例子是使用 VC6 生成的一个 Hello World 控制台程序，及其相应的 Makefile 编写方法。

在 VC6 的命令提示符下，生成 Release 版的命令为：

```
nmake clean cfg=Release
```

生成 Debug 版的命令为：

```
nmake clean cfg=Debug
```

现在，你已经具有编写简单 Makefile 的能力了。