# 程序的编译与链接理解

一般来说，无论是C、C++、还是pas，首先要把源文件编译成中间代码文件，在Windows下也就是 .obj 文件，UNIX下是 .o 文件，即 Object File，这个动作叫做编译（compile）。然后再把大量的Object File合成执行文件，这个动作叫作链接（link）。即：

源文件编译成中间文件🡪 编译。

将可执行文件合成可执行文件 🡪 链接。

编译时，编译器需要的是语法的正确，函数与变量的声明的正确。对于后者，通常是你需要告诉编译器头文件的所在位置（头文件中应该只是声明，而定义应该放在C/C++文件中），只要所有的语法正确，编译器就可以编译出中间目标文件。

链接时，主要是链接函数和全局变量，所以，我们可以使用这些中间目标文件（O文件或是OBJ文件）来链接我们的应用程序。链接器并不管函数所在的源文件，只管函数的中间目标文件（Object File），在大多数时候，由于源文件太多，编译生成的中间目标文件太多，而在链接时需要明显地指出中间目标文件名，这对于编译很不方便，所以，我们要给中间目标文件打个包，在Windows下这种包叫“库文件”（Library File)，也就是 .lib 文件，在UNIX下，是Archive File，也就是 .a 文件。

总结：源文件首先会生成中间目标文件，再由中间目标文件生成执行文件。在编译时，编译器只检测程序语法，和函数、变量是否被声明。如果函数未被声明，编译器会给出一个警告，但可以生成Object File。而在链接程序时，链接器会在所有的Object File中找寻函数的实现，如果找不到，那到就会报链接错误码（Linker Error），在VC下，这种错误一般是：Link 2001错误，意思说是说，链接器未能找到函数的实现。你需要指定函数的Object File.

# Makefile 介绍

make命令执行时，需要一个 Makefile 文件，以告诉make命令需要怎么样的去编译和链接程序。

首先，我们用一个示例来说明Makefile的书写规则。以便给大家一个感兴认识。这个示例来源于GNU的make使用手册，在这个示例中，我们的工程有8个C文件，和3个头文件，我们要写一个Makefile来告诉make命令如何编译和链接这几个文件。我们的规则是：

1）如果这个工程没有编译过，那么我们的所有C文件都要编译并被链接。

2）如果这个工程的某几个C文件被修改，那么我们只编译被修改的C文件，并链接目标程序。

3）如果这个工程的头文件被改变了，那么我们需要编译引用了这几个头文件的C文件，并链接目标程序。

只要我们的Makefile写得够好，所有的这一切，我们只用一个make命令就可以完成，make命令会自动智能地根据当前的文件修改的情况来确定哪些文件需要重编译，从而自己编译所需要的文件和链接目标程序。

# Makefile的规则

在讲述这个Makefile之前，还是让我们先来粗略地看一看Makefile的规则。

target ... : prerequisites ...

command

...

...

target也就是一个目标文件，可以是Object File，也可以是执行文件。还可以是一个标签（Label），对于标签这种特性，在后续的“伪目标”章节中会有叙述。

prerequisites就是，要生成那个target所需要的文件或是目标。

command也就是make需要执行的命令。（任意的Shell命令）

这是一个文件的依赖关系，也就是说，target这一个或多个的目标文件依赖于prerequisites中的文件，其生成规则定义在command中。说白一点就是说，prerequisites中如果有一个以上的文件比target文件要新的话，command所定义的命令就会被执行。这就是Makefile的规则。也就是Makefile中最核心的内容。

## 示例