## 编译器的工作过程

源码要运行必须先转成二进制的机器码，这是编译器的任务。

下面针对gcc编译器进行讲解

#### 第一步 配置

编译器在工作之前需要知道当前系统环境，比如标准库在哪里、软件的安装位置在哪里、需要安装哪些组件等等。这个确定编译器参数的步骤，就叫做“配置”。

这些配置信息保存在一个配置文件中，约定俗成是一个叫做configuration的脚本文件。通常它是由autoconf生成的

#### 第二步 确定标准库和头文件的位置

要通过配置文件告诉编译器源码要用到的标准库函数和头文件

#### 第三步 确定依赖关系

对于大型项目来说，源文件之间往往存在依赖关系，编译器需要确定编译的先后顺序。假定A文件依赖于B文件，编译器应该做到下面两点：

1. 只有在B文件编译完成后，才开始编译A文件
2. 当B文件发生变化时，A文件会被重新编译

编译顺序保存在一个叫做makefile的文件中，而makefile文件由configuration脚本运行生成，这就是为什么编译时configuration必须首先运行的原因。

#### 第四步 头文件的预编译

不同的源码文件可能用到同一个头文件。为了节省时间，编译器会在编译源码之前，先编译头文件。这保证了头文件只需编译一次，不必每次用到的时候重新编译。并不是头文件的所有内容都会被预编译。用来声明宏的#define命令，就不会被编译。

#### 第五步 预处理

预编译完成后，编译器就开始替换掉源码中的头文件和宏。这一步称为“预处理”，因为完成之后，就要完成真正的处理了。

#### 第六步 编译

预处理之后，编译器就开始生成机器码。对于某些编译器来说，还存在一个中间步骤，会先把源码转为汇编代码，然后再把汇编代码转为机器码。这种转码后的文件称为对象文件(object file)。

#### 第七步 连接

对象文件还不能运行，必须进一步转成可执行文件。比如代码中会有c语言标准库提供的函数。那么编译器的下一步工作，就是把外部函数的代码(通常是后缀名为.lib和.a的文件)，添加到可执行文件中。这就叫做连接。这种通过拷贝，将外部函数库添加到可执行文件的方式，叫做静态连接。

Make的作用就是从第四步头文件预编译开始，一直到做完这一步。

#### 第八步 安装

上一步的连接是在内存中进行的，即编译器在内存中生成了可执行文件。下一步，将可执行文件保存到用户事先执行的安装目录。

实际上，这一步还必须完成创建目录、保存文件、设置权限等步骤。

#### 第九步 操作系统连接

要求在操作系统中登记这个程序的元数据：文件名、文件描述、关联后缀名等等。

#### 第十步 生成安装包

通常是将可执行文件(连带相关的数据文件)，以某种目录结构保存成压缩文件包，交给用户。

#### 第十一步 动态链接

前面已经说过，静态连接就是将外部函数库拷贝到可执行文件中。这样做的好处是，适用范围比较广，不用担心用户机器缺少某个库文件；缺点是安装包会比较大，而且多个应用程序之间无法共享库文件。动态链接的做法正好相反，外部函数库不进入安装包，只在运行时动态引用。好处是安装包会比较小，多个应用程序可以共享库文件；缺点是用户必须安装好库文件，而且版本和安装位置都必须符合要求，否则就不能正确运行。