编译器的工作过程（以gcc编译器为例）

大家都知道，我们用c、c++写出来的程序计算机是看不懂的，计算机只能看懂由0和1组成的机器码。这个将高级语言翻译成机器语言的过程由谁来做呢？编译器。编译器的作用就是将源码翻译成计算机可以看懂的语言，并且生成计算机可以执行的程序。下面我主要介绍一下gcc(GNU Compiler Collection, GNU编译器套件, 可以编译包括c、c++、Fortran、Pascal、Objective-C、Java、Ada和Go在内的多种编程语言)的工作过程。

1、配置

编译器要得知当前的系统环境，例如你所用语言的标准库在哪里、生成的软件要安装在哪里等等。不同计算机的环境是不同的，因此编译器需要一份配置文件来获得编译参数，从而适应不同的计算机环境，从而编译出在不同计算机上都可以正常运行的代码。通常这份配置文件是由一个叫做Autoconf的软件自动生成的一个叫做configure的脚本文件，gcc通过运行这份脚本文件来获知编译参数。

2、确定标准库和头文件的位置

对于源码中用到标准库函数和头文件的情况（实际上几乎所有程序都会用到），编译器需要通过上一步生成的配置文件来获知标准库和头文件的位置。通常来说配置文件里会列出一些具体的目录，等到编译时，编译器就会到这几个目录中寻找目标。

3、确定依赖关系

文件之间往往不是独立的，而是相互依赖的。比如A文件的内容要依赖于B文件的内容（例如A文件内写有#include “B.h”）。这时候编译器就要确定编译这两个文件的先后顺序。在这种A依赖于B的情况下，编译器就要确认做到以下两点：

(1)只有在B编译完成之后，才开始编译A；

(2)每当B变化时，便重新编译A。

这种编译顺序记录在一个叫做makefile的文件里，而这个文件是在运行上文提到的configure脚本时生成的。因此在编译时，必须首先运行configure脚本文件。

4、预编译（precompilation）

我们写程序时经常会有多个文件都引用了同一个文件的情况。在这种情况下，编译器为了防止这个被引用的文件被编译多次，就会在编译源文件之前，先将全部头文件编译完。这个过程就叫做编译器的预编译。

5、预处理（preprocessing）

在这一步中，编译器会将头文件和宏替换为其具体指代的内容，还会移除代码中的注释。例如在如下代码中：

```
A.h:
...
int testA(int a, string b);

B.c:
#include “A.h"

int main()
{
    testA(3, “test”);                    //aaaaaaaaaa
```

}

则经过预处理后，B文件会变成下面的样子：

B.c
```
int testA(int a, string b);

int main()
{
    testA(3, “test”);
}
```

6、编译（Compilation）
到了这一步，编译器就会将预处理后的源码翻译成机器码。某些编译器会先将源码翻译成汇编语言，然后再翻译成机器码。转码后的文件称为对象文件。
7、链接（Linking）
这里说的链接是指静态链接，以区别于在运行时发生的动态链接（和编译器无关）。这一步的作用是将需要的外部函数的代码（通常是后缀名为.lib和.a的文件）添加到可执行文件中。
8、安装（Installation）
上一步的链接是在内存中生成了可执行文件，这一步的安装就是将可执行文件保存到用户事先规定好的目录中的过程。
9、操作系统链接
这一步的目的是通知操作系统，刚刚安装好的可执行文件已经可以被使用了。编译器通知操作系统的方式便是在文件中保存一下这个程序的元数据，例如文件名、文件描述、关联后缀名等等。例如在Linux系统下，这些信息通常保存在/usr/share/applications目录下的.desktop文件中。
10、生成安装包
到了这一步，编译器的任务就是将可执行文件和相关的数据文件等以某种数据打包好，然后交给用户。

至此，编译器的任务就告一段落了。不过上面提到了静态链接，那么这里我还想再提一下与之相对应的动态链接。
静态链接的特点是在运行前就将所有需要的外部函数库都链接到可执行文件内，这样做的好处是不管你将这个程序移植到哪里运行，都不用担心缺少某个必要的库文件；但是缺点也很明显，就是安装包的体积势必会比较大。而动态链接则是在运行时引用外部库文件，而不让库文件进入安装包。动态链接的优点就是安装包会很小，但缺点就是使用这个程序的机器上必须要安装好所需的库文件。

By Caesar