言。例如，C 编译器和 Fortran 编译器产生的输出文件用的都是一样的汇编语言。

- 汇编阶段。接下来，汇编器(as)将 hello.s 翻译成机器语言指令，把这些指令打包成一种叫做可重定位目标程序(relocatable object program)的格式，并将结果保存在目标文件 hello.o 中。hello.o 文件是一个二进制文件，它包含的 17 个字节是函数 main 的指令编码。如果我们在文本编辑器中打开 hello.o 文件，将看到一堆乱码。
- 链接阶段。请注意，hello 程序调用了 printf 函数，它是每个 C 编译器都提供的标准 C 库中的一个函数。printf 函数存在于一个名为 printf.o 的单独的预编译好了的目标文件中，而这个文件必须以某种方式合并到我们的 hello.o 程序中。链接器(ld)就负责处理这种合并。结果就得到 hello 文件，它是一个可执行目标文件(或者简称为可执行文件)，可以被加载到内存中，由系统执行。

旁注　GNU 项目

GCC 是 GNU(GNU 是 GNU's Not Unix 的缩写)项目开发出来的众多有用工具之一。GNU 项目是 1984 年由 Richard Stallman 发起的一个免税的慈善项目。该项目的目标非常宏大，就是开发出一个完整的类 Unix 的系统，其源代码能够不受限制地被修改和传播。GNU 项目已经开发出了一个包含 Unix 操作系统的所有主要部件的环境，但内核除外，内核是由 Linux 项目独立发展而来的。GNU 环境包括 EMACS 编辑器、GCC 编译器、GDB 调试器、汇编器、链接器、处理二进制文件的工具以及其他一些部件。GCC 编译器已经发展到支持许多不同的语言，能够为许多不同的机器生成代码。支持的语言包括 C、C++ 、Fortran、Java、Pascal、面向对象 C 语言(Objective-C)和 Ada。

GNU 项目取得了非凡的成绩，但是却常常被忽略。现代开放源码运动(通常和 Linux 联系在一起)的思想起源是 GNU 项目中自由软件(free software)的概念。(此处的 free 为自由言论(free speech)中的“自由”之意，而非免费啤酒(free beer)中的“免费”之意。)而且，Linux 如此受欢迎在很大程度上还要归功于 GNU 工具，它们给 Linux 内核提供了环境。

1.3　了解编译系统如何工作是大有益处的

对于像 hello.c 这样简单的程序，我们可以依靠编译系统生成正确有效的机器代码。但是，有一些重要的原因促使程序员必须知道编译系统是如何工作的。

- 优化程序性能。现代编译器都是成熟的工具，通常可以生成很好的代码。作为程序员，我们无须为了写出高效代码而去了解编译器的内部工作。但是，为了在 C 程序中做出好的编码选择，我们确实需要了解一些机器代码以及编译器将不同的 C 语句转化为机器代码的方式。比如，一个 switch 语句是否总是比一系列的 if-else 语句高效得多？一个函数调用的开销有多大？while 循环比 for 循环更有效吗？指针引用比数组索引更有效吗？为什么将循环求和的结果放到一个本地变量中，会比将其放到一个通过引用传递过来的参数中，运行起来快很多呢？为什么我们只是简单地重新排列一下算术表达式中的括号就能让函数运行得更快？

  在第 3 章中，我们将介绍 x86-64，最近几代 Linux、Macintosh 和 Windows 计算机的机器语言。我们会讲述编译器是怎样把不同的 C 语言结构翻译成这种机器语言的。在第 5 章中，你将学习如何通过简单转换 C 语言代码，帮助编译器更好地完成工作，从而调整 C 程序的性能。在第 6 章中，你将学习存储器系统的层次结构特性，C 语言编译器如何将数组存放在内存中，以及 C 程序又是如何能够利用这些知识从而更高效地运行。