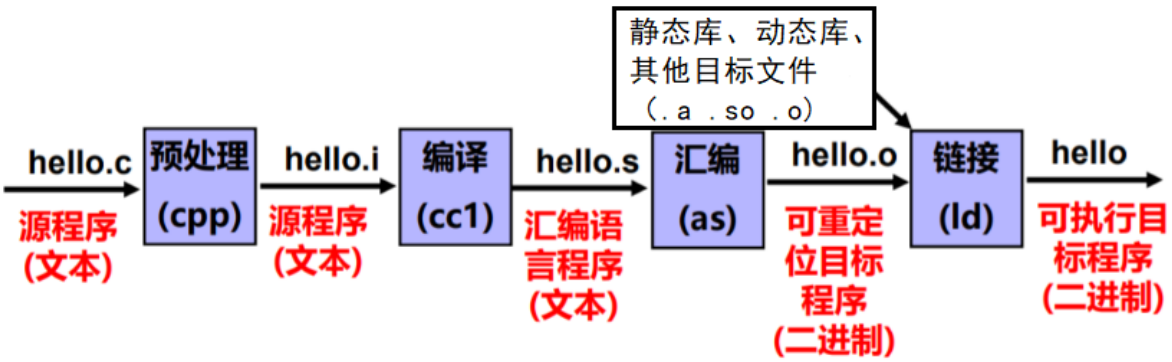
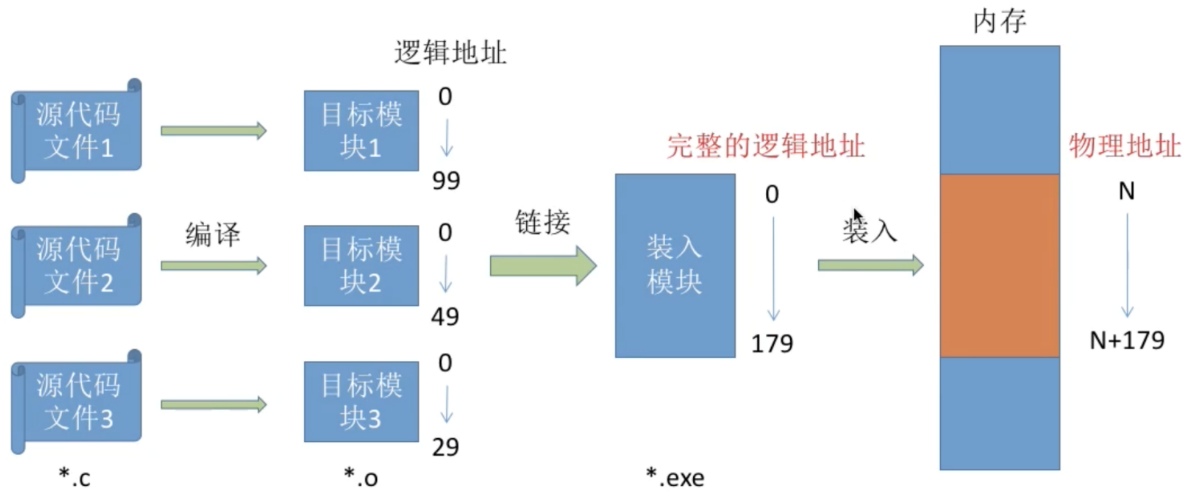
目标文件详解

源程序如何运行起来

通常说 C/C++ 源文件到可执行程序需要经过以下四个阶段：



最后链接出的 hello 也被称为目标文件，其由装入程序/加载器（loader）装入内存后就可以执行了。因此，完整的过程其实还应加上 **装入**：



目标文件

可以看出，在经过汇编器和连接器作用后，都会输出一个目标文件，那这两个目标文件有什么样的区别呢？说到这里我们先引入目标文件的形式。

目标文件的形式

1. 可重定位目标文件：包含二进制代码和数据，其形式可以和其他目标文件进行合并，创建一个可执行目标文
2. 可执行目标文件：包含二进制代码和数据，可直接被加载器加载执行
3. 共享目标文件：可被动态的加载和链接

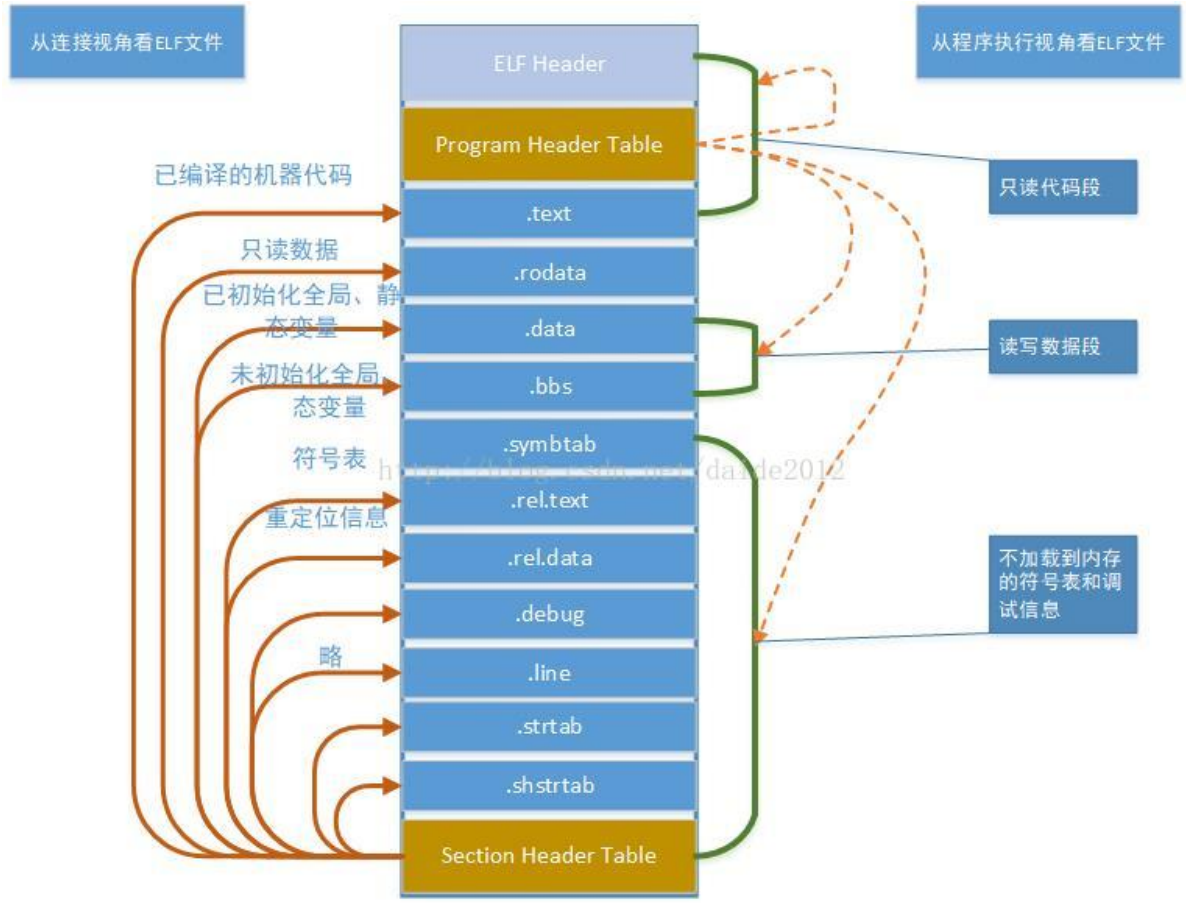
汇编器生成的就是可重定位目标文件，链接器作用后才生成可执行目标文件。链接器的作用就是以一组可重定位目标文件作为输入，生成可加载和运行的可执行目标文件，具体需要完成以下两个工作：

1. 符号解析：符号解析的目的是将目标文件中每个符号（静态变量、函数、全局变量）和其定义进行关联
2. 重定位：将每个符号的定义与具体在虚拟内存中的位置进行关联

ELF 文件

目标文件在不同的系统或平台上具有不同的格式，在 Unix 和 X86-64 Linux 上称为 ELF（Executable and Linkable Format）。

ELF 文件格式提供了两种不同的视角，在汇编器和链接器看来，ELF 文件是由 Section Header Table 描述的一系列 Section 的集合，而执行一个 ELF 文件时，在加载器（Loader）看来它是由 Program Header Table 描述的一系列 Segment 的集合。



ELF Header 描述了体系结构和操作系统等基本信息，并指出 Section Header Table 和 Program Header Table 在文件中的什么位置。

左边是从汇编器和链接器的视角来看这个文件：Section Header Table 中保存了所有 Section 的描述信息。Program Header Table 在汇编和链接过程中没有用到，所以是可有可无的。

右边是从加载器的视角来看这个文件：Program Header Table 中保存了所有 Segment 的描述信息；Section Header Table 在加载过程中没有用到，所以是可有可无的。

> Section Header Table 和Program Header Table 并不是一定要位于文件开头和结尾的，其位置由 ELF Header 指出，上图这么画只是为了清晰。

我们在汇编程序中用 .section 声明的 Section 会成为目标文件中的 Section，此外汇编器还会自动添加一些 Section（比如符号表）。Segment 是指在程序运行时加载到内存的具有相同属性的区域，由一个或多个 Section组成，比如有两个 Section 都要求加载到内存后可读可写，就属于同一个 Segment。有些 Section 只对汇编器和链接器有意义，在运行时用不到，也不需要加载到内存，那么就不属于任何 Segment。

可重定位目标文件需要链接器做进一步处理，所以一定有 Section Header Table；可执行文件需要加载运行，所以一定有 Program Header Table；而共享库既要加载运行，又要在加载时做动态链接，所以既有 Section Header Table 又有 Program Header Table。

更详细的分析见：[ELF文件详解—初步认识daide2012的博客-CSDN博客elf文件]

参考资料

1. 操作系统 程序如何运行：编译、链接、装入 baiiu-CSDN博客
2. ELF文件详解—初步认识*daide2012的博客-CSDN博客*elf文件