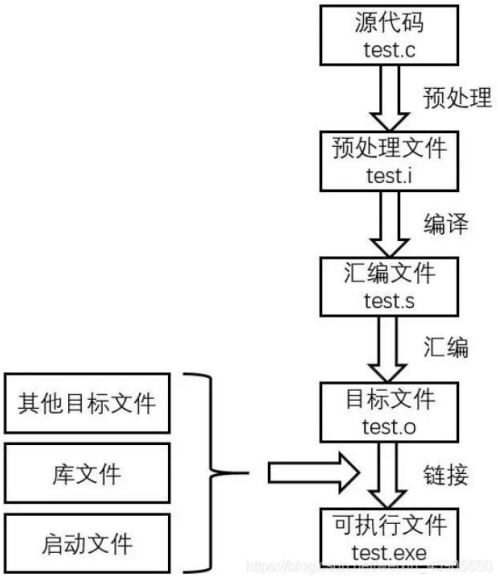
https://blog.csdn.net/weixin_45905650/article/details/107916093

编译器

把源代码转换成（翻译）低级语言的程序，一个现代编译器的主要工作流程：源代码 (source code) → 预处理器 (preprocessor) → 编译器 (compiler) → 目标代码 (object code) → 链接器 (Linker) → 可执行程序 (executables)。

个人理解，编译器负责将高级语言，如C、C++、Pascal/Object Pascal（Delphi）、Golang等等这些编译型语言转换成汇编语言，然后汇编器和链接器负责将汇编语言变为机器可识别的二进制文件。网上有些资料会说编译就是将高级语言或者汇编语言变成机器识别的二进制语言，个人理解这是因为一般的编译器都将汇编器链接器包含在内，所以会这样说。

下面给出C程序编译的过程，帮助大家理解：



解释器

直接把高级编程语言一行一行转译运行，重点是不会一次把整个程序转译出来，因此运行速度比较缓慢，它每转译一行程序就立刻运行，然后再转译下一行，再运行，如此不停地进行下去。

个人理解，解释器是帮助解释型语言一行一行的翻译成机器语言，程序是不需要编译的，程序在运行时才翻译成机器语言。比如Java、Perl、Python、basic、C#、PHP、Ruby、MATLAB等等。脚本语言也是解释型语言，比如VBScript、JavaScript、installshield script、ActionScript等等，脚本语言不需要编译，可以直接用，由解释器来负责解释。

https://www.cnblogs.com/clemente/p/10413618.html

编译器与解释器的区别和工作原理

这篇教程，我们一起来了解编译器和解释器的工作原理，这对我们提升能力很重要。

我一直认为概念或理论知识是非常重要的，让我们能够将很多模糊的认知关联起来，变得更加清楚，从而打开视野，上升到新的层次。

但是，一般来说，在刚刚入门的时候，接触一些概念性、理论性的内容，不但非常枯燥，而且难以理解。

而在一定时间的学习接触之后，再来看这些东西，则会变得更加容易领悟，理解的更透彻。

这篇教程会包含很多专业术语，我会对其中一些专业术语进行解释，对于一些未做解释的专业术语，建议大家通过搜索引擎进行理解。

首先，从Python这种编程语言说起。

它有以下几个特点：

- 面向对象：在本站的《Python3萌新入门笔记》中有专门的文章，简单来说是指在程序设计中能够采用封装、继承、多态的设计方法。
- 动态语言：是在运行时可以改变其结构的语言；例如，在程序运行过程中，给一个类的对象添加原本不存在的属性。
- 动态数据类型：变量不需要指定类型，但需要解释器执行代码时去辨别数据类型；这个特点让编程变得简单，但代码执行效率变低。
- 高级语言：是指高度封装了的编程语言，相对于机器语言，更加适合人类编写与阅读。
- 解释型语言：是指无需编译，直接能够将源代码解释为机器语言进行运行的语言。

从最后一个特点，我们能够看到Python是解释型语言，也就是说源代码需要通过解释器进行解释执行。

编程语言分为编译型语言和解释型语言，我们需要了解它们的区别，才能够更好的理解编译器和解释器的区别。

相信大家都知道C和C++。

这两种语言都是编译型语言。

编译型语言的特点是执行速度快，缺点是什么呢？

编译型语言需要编译器处理，主要工作流程如下：

源代码 (source code) → 预处理器 (preprocessor) → 编译器 (compiler) → 目标代码 (object code) → 链接器 (Linker) → 可执行程序 (executables)

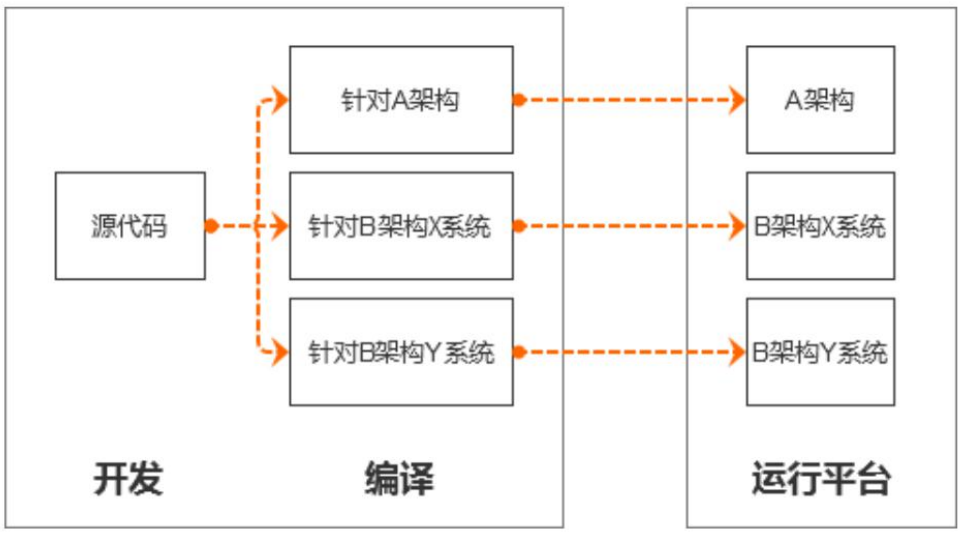
在这个工作流程中，编译器调用预处理器进行相关处理，将源代码进行优化转换（包括清除注释、宏定义、包含文件和条件编译），然后，通过将经过预处理的源代码编译成目标代码（二进制机器语言），再通过调用链接器外加库文件（例如操作系统提供的API），从而形成可执行程序，让机器能够执行。

在这个工作流程中，目标代码要和机器的CPU架构相匹配，库文件要和操作系统相匹配。

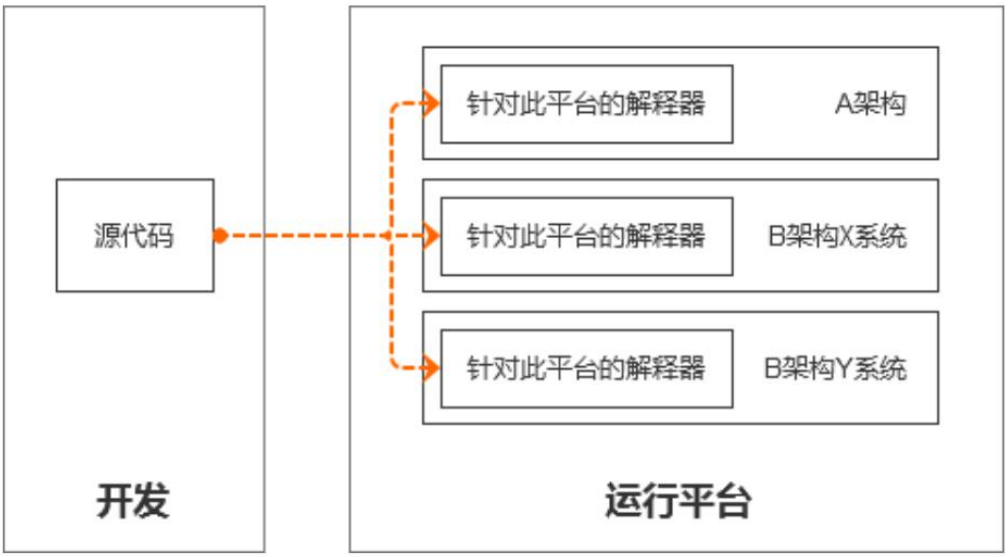
如果想在不同CPU的机器或者系统上运行C语言的源代码，就需要针对不同的CPU架构和操作系统进行编译，这样才能够在机器上运行程序。

所以，编译型语言的缺点我们就看到了，它不适合跨平台。

而且，到这里大家应该能知道，为什么CPU一样，但是exe程序只能Windows中运行，而不能在Mac中运行了。



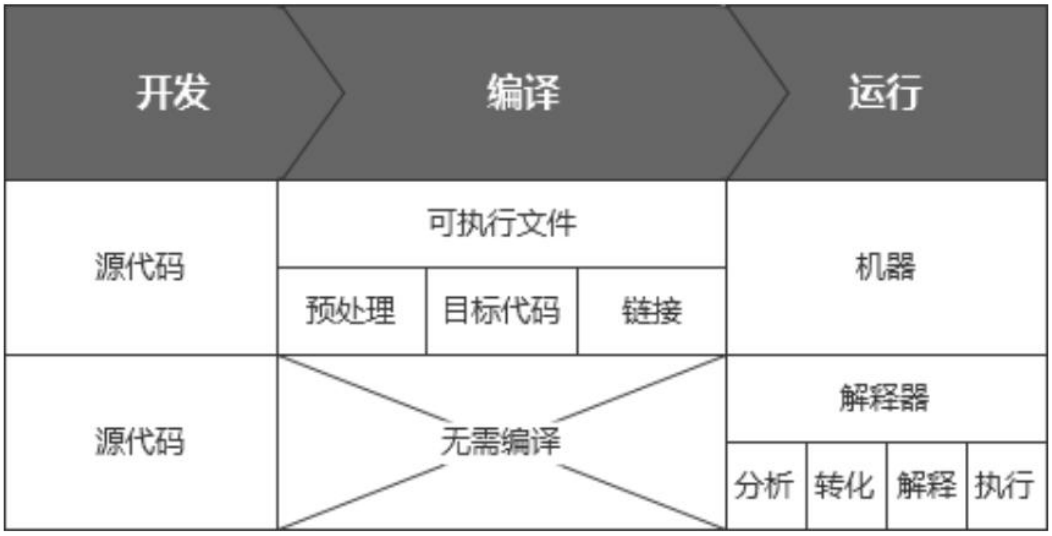
早期的解释器就是这样的工作流程：源代码 (source code) → 解释器 (interpreter) 。



源代码无需预先编译成可执行程序。

在程序执行时，解释器读取一句源代码之后，先进行词法分析和语法分析，再将源代码转换为解释器能够执行的中间代码（字节码），最后，由解释器将中间代码解释为可执行的机器指令。

所以，编译型语言的可执行程序产生的是直接执行机器指令，而解释型语言的每一句源代码都要经过解释器解释为可以执行的机器指令，相比之下解释型语言的执行效率会低一些。



但是，解释型语言在不同的平台有不同的解释器，源代码跨平台的目的实现了，开发人员不用再考虑每个平台如何去编译，只需要关注代码的编写，编写完的代码在任何平台都能无需修改（或少量修改）就能正确执行。

例如，Linux系统中执行Python源代码时支持 fork()函数，而window系统中不支持这个函数，如果将运行在Linux系统中的源代码移植到Windows系统，这时就需要进行修改。