Understand C Programming Language

Under C Programming Language

3230160@zju.edu.cn

前言

朋友们，“记住”与“记起”之间距离甚大 / 如同从卢加到绸缎巴乌特的国家

——安娜·阿赫玛托娃

当读者阅读到这一页时，不妨考虑一下上面引用的诗句，最好利用高考留给你的绝佳瞬时记忆把它背下来，然后再阅读下文。

你玩过 Minecraft 吗？玩的是 Java 版还是基岩版？如果你玩的是 Java 版，可别忘了在新电脑上安装 Java 环境。如果你玩的是基岩版，那也不是很费劲：你不需要安装 Java 环境了，直接从应用市场下载安装就行。唯一受伤的是平时游玩基岩版、但刚刚开始使用 Linux 操作系统的玩家：他们意外地发现，自己最习惯玩的基岩版不支持这个系统，只能投奔 Java 阵营。

对于大部分玩家而言，这两个版本的 Minecraft 的区别微不可察。最早的 Minecraft 只有 Java 版，而基岩版是在微软收购其开发商 Mojang 之后开发的新版本，最大的区别在于 Minecraft 基岩版采用 C++编写，而 Minecraft Java 版采用 Java 编写。微软说，基岩版的性能比 Java 版好得多，而且由于不用安装 Java 环境，玩家的体验好上不少。但是，基岩版是针对平台编译的，资本主义的化身微软肯定不会顾及人少事多的 Linux 用户，而只向有限的平台推出适配；而 Java 版则是“一次编译、处处执行”，只需要编译一次，就天然适配了所有平台……

——等等、等等！你说的这些 Java 啊、C++啊、编译啊、执行啊，都是什么玩意？

突击检查！上面要求你背下来的内容，你背下来了吗？如果我们将“记住”当成是我们向计算机保存内容，包含着我们主动而计算机被动的意味，那么“记起”就是计算机向我们展示我们所保存进去的内容，主动和被动的关系一下调转了过来。一个关键的问题在于，我们无法和计算机直接交流。**对于一块既要被“记住”又要被“记起”的信息**，一定要经过某种形式的翻译，将其转换为二进制编码，才能被计算机接受。这种过程就是编码。编译就是一种编码的过程，它面向的信息就是程序源代码。

——同样是编码为计算机可以理解的形式，为什么 Java 版和 C++版有区别呢？

好问题。一方面，这两种编程语言提供的功能不同，要实现同一个特性，就得采用不同的写法，使这些功能配合起来。但是，每一种功能的编码方式也有所不同，这就使得最终实现相同的特性，却对应了不同的二进制编码；另一方面，C++是编译型语言，而 Java 是解释型语言，它们之间的区别就是用这种语言编出的程序需不需要一个额外的“解释器”。对于 C++程序而言，它经过一次编译就转换成了计算机能够识读的机器语言；而 Java 程序的编译过程则是将其转换为一种计算机无法直接读懂的“中间语言表示”（IR），再借用解释器将其解释为计算机能够读懂的指令。因此，Java 解释器（也就是 JVM 虚拟机）承担了二次转译过程，而抹平了不同平台上对于二进制编码的偏好，统一成一种语言表示。

——就像是 DNA、mRNA 和蛋白质一样！

对，也不对。mRNA 是指导蛋白质合成的直接模板，而中间语言表示一般不会被翻译为二进制**可执行文件**，而是指导解释器里已经编译好的二进制指令按它所描述的规则执行。硬要说的话，更像是 rRNA 等与蛋白质共同发挥作用的结构性 RNA。当然，我们同样可以指导 IR“合成”程序，比如说我们之后要介绍的 clang/LLVM 编译工具链，它能将编译型语言 C 语言源代码先转化为 LLVM IR，再从 LLVM IR 编译为**可执行文件**。我们将在这套工具链上学习 C 语言。同样地，基于 JVM 的 Kotlin、基于.NET Framework（另一种解释器）的 C# 等等也能跳过解释环节直接编译为二进制**可执行文件**。

——可执行文件？那也就有“不可执行文件”咯？

有，也没有。对于一台计算机而言，任何信息都不过是 0 与 1 的排列组合罢了，所以，如果我们将这篇文档看作可执行文件而将其加载到内存中，说不定真能凑出几行指令而被执行。另一方面，如果我们编写一个配合于这篇文档的内容的解释器（比如读到括号里的字就跳转到上一行啦、读到句号就输出啦……[1]），这篇文档也可以是一段可执行文件。但是，对于大多数操作系统而言，它们有一套专门的算法识别一段数据是否可执行，这种算法一般是靠检测文件的结构来判断它是否可执行，因为可执行文件具有特殊的结构。如果一段数据是可执行文件并被要求执行，那么它就会被系统从硬盘加载到内存中，再经过一系列复杂的准备，让处理器和内存进入状态。这样之后，程序才会被执行。

——也就是说，程序是由别的程序加载的咯？那么最早被加载的程序是怎么被加载的呢？

这是个“鸡生蛋、蛋生鸡”的问题，但是谁说程序只能是软件？计算机上电后，硬件电路会将硬盘上的某个地址上特定大小的数据加载到内存中，然后将指令指针指向数据的开头。那一段被硬件加载上去的数据，接着将其他程序加载到内存中并开始执行。[2]

——这一切又和 C 语言有什么关系呢？

C 语言只是千千万万种编程语言中的一种，而编程语言和程序设计的学问也只是计算机科学的沧海一粟。在笔者看来，计算机科学最美好的一点就是它可以超脱物质世界的桎梏，无论是烂漫的幻想还是严肃的论证，都能够在计算机的虚拟当中，找到它们现实的样子。对于大部分学习这门课程的同学而言，他们不会从事计算机相关的职业，甚至包括笔者中的大多数人也是这样。但是，C 语言作为解开桎梏的第一把钥匙，将留给我们的是无尽的自由。

[1] 可参见我校 2024 暑学期课程《安全攻防实践》的一道题：https://courses.zjusec.com/topic/rev-lab3/#challenge-1-ichicken-60，程序本体是一个文本文档，却同样可以“执行”

[2] 当然，不同架构的芯片和不同的操作系统在加载这一段程序前后的策略上有所不同。参见：《x86 汇编语言：从实模式到保护模式》《Linux 内核完全注释》以及白洪欢老师《汇编语言》课程的讲义。