Java替代C语言的可能性 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/145/2021_2022_Java_E6_9B_BF_E4_BB_A3_c104_145039.htm 前不久CSDN刊登了一篇《C语言已经死了》的文章，引起了一些争论。其实那篇文章是从Ed Burnette的博客上转载来的，原文题目是“ Die， C， die ！ ”，直译过来应该是《去死吧，C！》，表达的是一种诅咒，而不是判断。翻译称《C语言已经死了》，显然是一种煽风点火的误读。CSDN网友对于其观点已经进行了批判，不过坦率地说，由于这些批判基于一个扭曲的翻译文本，所以不但没有什么新鲜的地方，而且也没有抓住原作者的重点。 实际情况是这样的，最近一段时间，在国外的技术社群里刮起了一股风，不少人在讨论Java做为C语言替代者而成为最主流的基础软件编程语言的可能性。从大部分人发表的观点来看，对于Java替代C的趋势还是支持的。 基础软件是指这样一类软件，其主要任务是把计算机的潜能充分发挥出来，面向上层应用软件提供一个高效、可靠的功能集。这些软件会被密集地调用，性能上的一点点滞后都会在实践中被成百上千倍的放大。所以对于基础软件来说，性能至少与可靠性一样重要。我们在一些基础软件的源代码里，常常看到一些丑陋的设计，看到一些变态的黑客技巧，在其他的领域里，这是不被鼓励的，但是在基础软件中，这就是合理的，可以接受的。 C语言目前仍在一些领域里坚挺，在操作系统、虚拟机和设备驱动程序开发方面，它可能是永远的王者。但是在其他的基础软件领域，比如数据库、网络服务器、图形图像处理等，C语言继续占据霸主地位的原因其实只有两个，一

是快，二是熟悉的人多，而且经验丰富。 但是这两点现在都遭到了挑战。 首先是速度。Java的执行速度在JDK1.4的时候达到了这样一个水平，就是对于一个一般水平的开发者来说，他写的C 程序已经不再比对等的Java 程序跑得更快了。随后的JDK 5.0和6.0进一步提高了执行性能，由不同的组织举行的多项评测结果表明，Java与C语言的整体执行效率差距在一倍以内，也就是说，素以速度著称、并且为了速度放弃了很多东西的C语言，现在比装备齐全的Java只快不到一倍了。这还不算，如果考虑到新的计算环境，C语言的速度优势有可能仅仅是一个错觉。因为，世界上只有很少的人有能力在多CPU计算平台上用C语言写出又快又正确的大程序，在这些人中间，又只有很少很少的人有能力用C语言写出一个在大型的、异构的网络环境下能够充分发挥各节点计算能力的大规模并行程序。也就是说，你也许有能力把程序效能提高一倍，从而充分发挥一台价值6000元人民币的PC的计算潜力，为客户节省1000元钱。但如果是在一个由1000台机器组成的大型异构网络并行计算的环境下，你写的C程序恐怕性能还会远远低于对应的Java程序，更不要说巨大的后期维护成本，而由此带来的损失可能是1000万或者更多。 其次是经验。很多人都宣称自己的C功力如何如何了得，但是实际上，即使是真正的C高手也不得不花相当可观的时间来寻找并且调试错误，尤其是内存方面的错误。大部分用C写的上规模的软件都存在一些内存方面的错误，需要花费大量的精力和时间把产品稳定下来。这还没有把安全方面的缺陷考虑在内，现在大部分的开发者在代码安全方面的知识都很薄弱，安全漏洞在代码中相当普遍，而在C语言中，这一不足暴露得格

外明显。最大的挑战或许得说是并发问题了，并发是一个很复杂的问题，需要在相当高的抽象层面上解决，而C语言的抽象机制过于简单，提供不了高层的抽象，因此在开发者只能从一些“并发原语”出发去构造并发程序，这跟用铅笔刀锯大树没什么分别，直截了当地说，大部分C程序员根本没有能力编写高效无缺陷的并发程序。 所以残酷的事实是，当一个人说自己的C语言如何了得，经验如何丰富时，非常可能他说的是，自己在用C语言写单机、单线程的，不会遭到外界攻击的，在时间预算上没有什么压力，而且用户能够忍受一个很长的产品稳定期的应用程序方面非常有经验。遗憾的是，市场环境和计算环境已经完全变化。面对更复杂的计算环境，用C语言来编写高质量的大规模软件，是只有真正的专家团队才能完成的工作。如果你曾经有过连续数日苦苦追踪和调试一个内存泄露、或者线程错误的经历，你就会明白，你可能不是这样的专家。 相比之下，Java在抽象机制、基础设施、安全和并发方面，与C语言比起来，就好像是马克沁重机枪对弓箭。比如并发，Java 5.0加入的java.util.concurrent包，可能是目前主流语言中对于并发问题最强有力的支持库。Java的内存管理和安全机制，也已经被实践证明确实能够有效地减少程序的缺陷。这也就是那篇诅咒文章的原文的意图。 所以，我的态度明确的，我认为Java替代C是一个进步的想法，不过世界上进步的想法很多，能够美梦成真的却寥寥无几。Java是否真的能够在基础软件领域强有力地替代C语言呢？我看至少短期内还做不到，原因如下：1. 人的问题。能够用C语言写出优秀基础软件的人固然不多，能用Java写出来的人恐怕更少。Java有好几百万开发者，

但是他们在干什么？大部分是去搞企业级开发、Web开发了，有多少人真的理解Java的内存模型？有多少人能够熟练使用concurrent包中提供的那些工具？很多使用Java多年的人没有写过socket程序，不了解Java多线程的开销，不清楚如何进行性能诊断和调优，而这些在写基础软件的时候是必备的技能。大部分Java程序员在刚刚学会Java之后就转向Web开发，把主要精力花费在掌握一个又一个大型的、复杂的、具有厚厚的抽象层和华丽结构的frameworks上，不但对真实计算机体系结构不清楚，对于Java虚拟出来的那个计算环境也不清楚。因此，要把Java社群编程转变成能够担负起下一代基础软件开发工作的尖兵，不但难度很大，而且必须花费足够的时间。2. Java的内存消耗太大。对于系统级程序来说，内存消耗大，就意味着cache命中率降低，与磁盘交换数据的可能性增大，对性能的影响还是比较严重的。现在很多人还是觉得Java慢，主要的原因已经不是Java跑得慢，而是由于内存消耗过大导致的综合性能下降。这个问题不解决，Java就只能用来做一些比较上层的基础软件。也许随着计算机硬件的发展，这个问题会逐步得到解决？ 3. 风格的问题。这个问题我认为是最严重的。基础软件开发崇尚的是自由、直接、透明、简单、高效，要像匕首一样锋利，像战士一样勇猛，像农夫一样朴实，反对繁琐华丽的设计，反对架床迭屋的层层抽象，反对复杂的结构和不必要的灵活性。而Java社群多年来形成的设计风格与此格格不入，甚至可以说是对立的。Java在意识形态上是要面向企业应用软件的开发，所以特别强调架构，强调设计模式，强调标准，强调规规矩矩，强调高姿态，强调一种华贵的宫廷气质。在C 中，你吃饭就是吃饭，捧起碗来喝酒，

放下筷子骂娘，甩开膀子抓肉，撸起袖子抹油。而在Java中，你经常为了要干某件事，先new一个对象，然后以这个对象为参数new另一个对象，如此这般重复n遍，得到真正需要的对象，最后就是为了调用那个对象的一个方法，就好比吃饭时焚香洗面，漱口净手，战战兢兢，毕恭毕敬。在C中，遇到问题要像亡命徒，像流氓版程咬金，管你三七二十一，冲上去就是三板斧，还怕劈不死你丫的。在Java里，遇到问题要像宋襄公，要张榜檄文，要名正言顺，要礼仪之邦，要把架子拉开了，把谱儿摆足了。Java的口号是，不管劈不劈的死，先把你小子感动了再说。这套繁琐的东西，对于基础软件开发来说，既不必要，也很难习惯。需要说明的是，这不是Java语言的问题，其实Java本身不必如此复杂、如此巴洛克。从语言本身来看，Java也可以是轻快直接的，也可是酣畅淋漓的。只不过十多年来几乎没有人这样用过，所以大家已经不知道：如果不来个一步三叩首，那么该怎么用Java写程序？ 正是因为上面的这种种原因（可能还不全面），直到最近，第一流的基础软件几乎都还是C语言编写的，或者至少其核心部分还是以C为主。而且我认为，在短期内，这种局面不会有大的改变。当然，如果Java社群能够克服上面的这些问题，充分发挥出Java本身的优势，在基础领域开发出一大批一流的支撑软件，那么局面是可以改变的，而且这种改变也是进步的，值得欢迎的。 