瘦身前后兼谈C 语言进化(1) PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/469/2021_2022__E7_98_A6_E8_BA_AB_E5_89_8D_E5_c67_469829.htm 一门语言应该是“ Make simple things simple, make complex things possible ”的。当我们用语言来表达思想的时候，这门语言应该能够提供这样的能力：即让我们能够最直接地表达我们的意思，多一分则太多，少一分则太少，好比古人形容美女：增一分则太肥，减一分则太瘦。 这个问题上，有一个我认为是广泛的误解，就是“ KISS便意味着要精简语言，并避免在编码中使用‘高阶’语言特性”。对此有一句话我觉得说得好：你不能通过从一门语言中去掉东西来增加表达力。高阶特性是一面利刃，用得不好固然伤了自己，但这并不表明就没有用。任何东西都是在它真正适用的地方适用，霸王硬上弓的话弓断弦崩反而伤及自身。所以，仅仅因为高阶特性容易误用（而且高阶特性的确也容易吸引人去用且容易误用，不过这是另一个问题），就断然在任何地方都不用并宣称这样才是KISS的话，便因噎废食了。举个例子，高阶函数是有用的，如果在真正需要高阶函数的地方不用高阶函数，那不是KISS，只能让解决方案（或者更确切地说，workaround）更复杂。lambda函数是有用的，但如果在真正需要lambda的地方不使用lambda，也只能导致更复杂更不直观的workarounds。OOP是有用的，但如果你的程序本来就只是简单的“ 数据*作 ”你偏要硬上OOP的话，不仅多了编码时间，而且还降低程序的可见度和可维护性，后者就意味着项目的money。拿C来说，这是一个广为诟病的问题。C 的偏向底层的应用领域

决定了有不少地方使用C 其实就是“ 数据 *作 ”，然而很多人却因为用的是C 编译器，便忍不住去使用高级特性，结果把本来简单的事情复杂化我自己就有不少次这样的经历：用了一大堆类之后，做完了回过头来再看，这些类都干嘛来着？需要吗？最关键的就是要清楚自己做的是什么事情，以及什么工具才是对你所做的事情最适合的。 说到这里不妨顺便说说另一个误解：“ 如果我反正用不着C 里面的高级特性，那还不如用C罢了 ”，鉴于C/C 的应用领域，的确有不少地方是可以用C 的C部分完成得很好的，所以这个误解被传播得还是蛮广泛的。这里的一个微妙的忽视在于：用C的话，你就用不到许多很好的C 库了。用C 的话，你完全可以在你自己的编码中不使用高阶特性（说实话，这需要清醒的头脑和丰富的经验，以及克制能力），但你还是可以利用众多的C 库来简化你的工作的：如果一个transform明明可以搞定的你偏要写一个for出来难道能叫KISS？如果一个vector就能避免绝大多数内存管理漏洞和简化内存管理工作你偏偏要手动malloc/free那能叫KISS（我见过不少用C 编码却到处都是malloc/free的）？如果最直接的方式是gc你偏偏要绕一大堆弯子才能保证正确释放那也不叫KISS（等C 09吧）。如果一个for_each(readdir_sequence(".", readdir_sequence::files), ::remove).能搞定的你偏要写：// in CDIR* dir = opendir(".").if(NULL != dir){struct dirent* de.for(. NULL != (de = readdir(dir)).){struct stat st.if(0 == stat(de->d_name, amp.amp. S_IFMT)){remove(de->d_name).}} closedir(dir).}