C语言最大难点揭秘[4] PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/246/2021_2022_C_E8_AF_AD_E8_A8_80_E6_9C_80_c97_246235.htm 使这些格式元素成为您日常工作的一部分。可以使用各种方法解决内存问题： 专用库 语言 软件工具 硬件检查器 在这整个领域中，我始终认为最有用并且投资回报率最大的是考虑改进源代码的风格。它不需要昂贵的代价或严格的形式；可以始终取消与内存无关的段的注释，但影响内存的定义当然需要显式注释。添加几个简单的单词可使内存结果更清楚，并且内存编程会得到改进。 我没有做受控实验来验证此风格的效果。如果您的经历与我一样，您将发现没有说明资源影响的策略简直无法忍受。这样做很简单，但带来的好处太多了。 检测 检测是编码标准的补充。二者各有裨益，但结合使用效果特别好。机灵的 C 或 C 专业人员甚至可以浏览不熟悉的源代码，并以极低的成本检测内存问题。通过少量的实践和适当的文本搜索，您能够快速验证平衡的 *alloc() 和 free() 或者 new 和 0delete 的源主体。人工查看此类内容通常会出现像清单 7 中一样的问题。 清单 7. 棘手的内存泄漏 static char *important_pointer = NULL. void f9() { if (!important_pointer) important_pointer = malloc(IMPORTANT_SIZE). ... if (condition) /* Ooops! We just lost the reference important_pointer already held. */ important_pointer = malloc(DIFFERENT_SIZE). ... } 如果 condition 为真，简单使用自动运行时工具不能检测发生的内存泄漏。仔细进行源分析可以从此类条件推理出证实正确的结论。我重复一下我写的关于风格的内容：尽管大量发布的

内存问题描述都强调工具和语言，对于我来说，最大的收获来自“ 软的 ”以开发人员为中心的流程变更。您在风格和检测上所做的任何改进都可以帮助您理解由自动化工具产生的诊断。 静态的自动语法分析 当然，并不是只有人类才能读取源代码。您还应使静态语法分析 成为开发流程的一部分。静态语法分析是 lint、严格编译 和几种商业产品执行的内容：扫描编译器接受的源文本和目标项，但这可能是错误的症状。 希望让您的代码无 lint。尽管 lint 已过时，并有一定的局限性，但是，没有使用它（或其较高级的后代）的许多程序员犯了很大的错误。通常情况下，您能够编写忽略 lint 的优秀的专业质量代码，但努力这样做的结果通常会发生重大错误。其中一些错误影响内存的正确性。与让客户首先发现内存错误的代价相比，即使对这种类别的产品支付最昂贵的许可费也失去了意义。清除源代码。现在，即使 lint 标记的编码可能向您提供所需的功能，但很可能存在更简单的方法，该方法可满足 lint，并且比较强键又可移植。