C语言编程常见问题解答之指针和内存分配 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/285/2021_2022_C_E8_AF_AD_E8_A8_80_E7_BC_96_c97_285300.htm 指针为C语言编程提供了强大的支持如果你能正确而灵活地利用指针，你就可以直接切入问题的核心，或者将程序分割成一个个片断。一个很好地利用了指针的程序会非常高效、简洁和精致。 利用指针你可以将数据写入内存中的任意位置，但是，一旦你的程序中有一个野指针("wild ” pointer)，即指向一个错误位置的指针，你的数据就危险了存放在堆中的数据可能会被破坏，用来管理堆的数据结构也可能会被破坏，甚至操作系统的数据也可能会被修改，有时，上述三种破坏情况会同时发生。 此后可能发生的事情取决于这样两点：第一，内存中的数据被破坏的程度有多大；第二，内存中的被破坏的部分还要被使用多少次。在有些情况下，一些函数(可能是内存分配函数、自定义函数或标准库函数)将立即(也可能稍晚一点)无法正常工作。在另外一些情况下，程序可能会终止运行并报告一条出错消息；或者程序可能会挂起；或者程序可能会陷入死循环；或者程序可能会产生错误的结果；或者程序看上去仍在正常运行，因为程序没有遭到本质的破坏。 值得注意的是，即使程序中已经发生了根本性的错误，程序有可能还会运行很长一段时间，然后才有明显的失常表现；或者，在调试时，程序的运行完全正常，只有在用户使用时，它才会失常。在C语言程序中，任何野指针或越界的数组下标(out-of-bounds array subscript)都可能使系统崩溃。两次释放内存的操作也会导致这种结果。你可能见过一些C程序员编

写的程序中有严重的错误，现在你能知道其中的部分原因了。 有些内存分配工具能帮助你发现内存分配中存在的问题，例如漏洞(leak，见7．21)，两次释放一个指针，野指针，越界下标，等等。但这些工具都是不通用的，它们只能在特定的操作系统中使用，甚至只能在特定版本的编译程序中使用。如果你找到了这样一种工具，最好试试看能不能用，因为它能为你节省许多时间，并能提高你的软件的质量。 指针的算术运算是C语言(以及它的衍生体，例如C )独有的功能。汇编语言允许你对地址进行运算，但这种运算不涉及数据类型。大多数高级语言根本就不允许你对指针进行任何操作，你只能看一看指针指向哪里。 C指针的算术运算类似于街道地址的运算。假设你生活在一个城市中，那里的每一个街区的所有街道都有地址。街道的一侧用连续的偶数作为地址，另一侧用连续的奇数作为地址。如果你想知道River Rd．街道158号北边第5家的地址，你不会把158和5相加，去找163号；你会先将5(你要往前数5家)乘以2(每家之间的地址间距)，再和158相加，去找River Rd．街道的168号。同样，如果一个指针指向地址158(十进制数)中的一个两字节短整型值，将该指针加3=5，结 果将是一个指向地址168(十进制数)中的短整型值的指针(见7．7和7．8中对指针加减运算的详细描述)。街道地址的运算只能在一个特定的街区中进行，同样，指针的算术运算也只能在一个特定的数组中进行。实际上，这并不是一种限制，因为指针的算术运算只有在一个特定的数组中进行才有意义。对指针的算术运算来说，一个数组并不必须是一个数组变量，例如函数malloc()或calloc()的返回值是一个指针，它指向一个在堆中申请到的数组。 指针的说明看起

来有些使人感到费解，请看下例： char *p ；上例中的说明表示，p是一个字符。符号“ * ”是指针运算符，也称间接引用运算符。当程序间接引用一个指针时，实际上是引用指针所指向的数据。 在大多数计算机中，指针只有一种，但在有些计算机中，指向数据和指向函数的指针可以是不同的，或者指向字节(如char。指针和void *指针)和指向字的指针可以是不同的。这一点对sizeof运算符没有什么影响。但是，有些C程序或程序员认为任何指针都会被存为一个int型的值，或者至少会被存为一个long型的值，这就无法保证了，尤其是在IBM PC兼容机上。 注意：以下讨论与Macintosh或UNIX程序员无关； 最初的IBM PC兼容机使用的处理器无法有效地处理超过16位的指针(人们对这种结论仍有争议。16位指针是偏移量，见9．3中对基地址和偏移量的讨论)。尽管最初的IBM PC机最终也能使用20位指针，但颇费周折。因此，从一开始，基于IBM兼容机的各种各样的软件就试图冲破这种限制。为了使20位指针能指向数据，你需要指示编译程序使用正确的存储模式，例如紧缩存储模式。在中存储模式下，你可以用20位指针指向函数。在大和巨存储模式下，用20位指针既可以指向数据，也可以指向函数。在任何一种存储模式下，你都可能需要用到far指针(见7．18和7．19)。 基于286的系统可以冲破20位指针的限制，但实现起来有些困难。从386开始，IBM兼容机就可以使用真正的32位地址了，例如象MS-Windows和OS／2这样一些操作系统就实现了这一点，但MSDOS仍未实现。 如果你的MSDOS程序用完了基本内存，你可能需要从扩充内存或扩展内存中分配更多的内存。许多版本的编译程序和函数库都提供了这种技术，但彼此之间

有所差别。这些技术基本上是不通用的，有些能在绝大多数MS-DOS和MS-WindowsC编译程序中使用，有些只能在少数特定的编译程序中使用，还有一些只能在特定的附加函数库的支持下使用。如果你手头有能提供这种技术的软件，你最好看一下它的文档，以了解更详细的信息。 7．1 什么是间接引用(indirection)? 对已说明的变量来说，变量名就是对变量值的直接引用。对指向变量或内存中的任何对象的指针来说，指针就是对对象值的间接引用。如果p是一个指针，p的值就是其对象的地址；*p表示“ 使间接引用运算符作用于p ”，*p的值就是p所指向的对象的值。 *p是一个左值，和变量一样，只要在*p的右边加上赋值运算符，就可改变*p的值。如果p是一个指向常量的指针，*p就是一个不能修改的左值，即它不能被放到赋值运算符的左边，请看下例： 例 7.1 一个间接引用的例子 #include int main() { int i. int * p . i = 5. p = amp.i) 后，打印i或*p的结果是相同的；你甚至可以给*p赋值，其结果就象你给i赋值一样。 