Universe Scientific Publishing
Computer System Networking and Telecommunications
计算机系统网络和电信，2019,1(2)
ISSN:2661-3719 (Print); 2661-3727 (Online)

C++ and Memory Management

Kexin MA Zhengde BAO Yawen TANG
School of Computer and Software, Jincheng College, Sichuan University, Chengdu,611731, China

Abstract

C++ with its unique packaging, inheritance, polymorphism and the development of procedures with its high efficiency, good performance and other advantages are used by many programmers. But the most cannot avoid part of the development of c + + program is a memory module, the use of memory directly reflects the running efficiency, but memory is different from visible and can modify the statement, scratching it can't see, do not pay attention to a little, will make the program cannot run for the problems, and to modify the fault, but it is not easy to detect where the mistakes. Based on the basic knowledge of C++ and memory, this paper briefly analyzes how to make the novice programmer better understand memory management.

Key Words

C++, Memory, Management

DOI:10.18686/jsjxt.v1i2.634

浅谈 C++与内存管理

马可心 鲍正德 唐娅雯
四川大学锦城学院计算机与软件学院，四川，成都，611731

摘 要

C++以其独有的封装、继承、多态特性以及用其开发的程序运行效率高、性能好等优点被许多程序员所使用。但开发 C++程序最不能避免的部分便是内存这一模块，内存的使用直接体现运行效率，但内存不同于看得见且能修改的语句，它看不见摸不着，稍微不注意，所出现的问题就会使程序不能运行，而要去修改这个错误的时候，却又不容易检测出哪里出现了错误。本文基于 C++与内存相关的基础知识，浅析了如何让初学 C++的程序员更加了解内存管理。

关键词

C++；内存；管理

1.引言

现在，越来越多的开发者选择 C++来进行计算机程序的开发。C++优越的性能让越来越多的人选择了这门编程语言。纵观 C++的学习，从基础的语法开始，已经或多或少的与内存产生了联系，而如果要进行更加复杂一点的程序的开发，内存，始终是不能逃避的话题。C++的开发，不同于其他面向对象语言，程序员需要非常谨慎地编写关于内存部分的语句，并且需要自己管理好内存。随着人们的需求在提升，现在的程序，需要更多的让语句与内存产生联系。

2.C++与内存联系

2.1 C++简单数据与内存

C++面向对象的思路，贯穿于整个所开发的程序之中。但在平时的使用之中，也会经常的使用到一些简单的数据，而这些数据，便可以称之为字符、数字等。通常是由程序员所定义的变量，向存储空间申请一片空间，然后将这些简单的数据进行存储。而这么一个过程，可以简略归纳为申请空间，存储数据。在使用简单的语句来完成的这一过程，并且有编译器根据语句来协助着开

发，使将要被存放的数据更稳定地存于其中，在之后的语句中进行使用。

但是，根据变量的不同，所申请的计算机内存位置也会相应的发生改变。按 C++知识，首先将变量按照代码块来进行划分为代码之外的变量，和代码内部的变量。而这样按代码块来分，则更轻易能明白到其机制。正常开发需要语句来让编译器去完成将数据存储和取出的操作，这一个操作需要重复，多次的完成，所以计算机系统为这一部分的数据在栈区上开辟了一个空间，以便操作。而代码外定义的变量则是被放在了静态存储区域。静态变量的空间在程序编译阶段进行分配，所分配内存在程序整个运行期间都存在。[1]代码外变量特点则是稳定，代码内如何运行与其无关。而相同特点的还有比较特殊的变量便是由 static 来进行定义的变量，如果定义在代码内，为静态局部变量，虽说是局部，但由于其特殊性，还是需要放在静态存储之中。

2.2 C++复杂数据与内存

C++由于他的三大特性被程序员所广泛的使用，而最能体现 C++面向对象思想的便是封装特性，所以，类与对象则是 C++中运用最多的方法，甚至可以说只要是大型项目都会看到类的踪影。类将私有成员、函数等封装起来，只提供接口，里面的数据所有人都不知晓其构造，但编程人员可以通过接口知道并使用。类一般用于比较复杂的数据，和数组一样，程序员更愿意将使用复杂、数量多的数据放在其中，而不是声明多个变量，这样可以提高开发的效率。在类中，根据属性可以定义成员，使用任意一种类型便可以定义以便来使用，同样，根据类型的不同，这些数据在内存中存储的位置也会不同。在类中，虽然会使用到普通的变量，但在开发的同时，也会在程序中加入到指针变量来指定数组、地址等。当利用 int 等传统的类型来定义时，数据会被存放在栈内存中，而在利用指针变量时，地址存在栈内存，但指向数据可以在堆也可以在栈中。指针变量与传统的变量存储方式相同，但指针却可以指向不同内存的数据。在类中，程序员通常需要用 new 来进行空间的申请，才能将数据平稳地存入到堆内存中，而在申请内存之前，声明的指针没有被使用到，便只会使用到指令代码区。

3.C++内存机制

C++内存的机制与 C 有些许不同，同时，这种不同也会映射到代码语句，熟悉内存机制会使程序员更加高效地开发关于内存方面的程序。

3.1 C++栈内存

栈内存是一个不那么灵活的区域，栈的分配以及栈的大小都由系统来管理，程序员控制不了这个区域。但正因为他不受程序员“控制”的特性，可使程序员用起来相当的放心。当声明传统的变量的时候，不需要向系统申请任何的空间，更不需要担心内存是否得到释放。这样的一种模式，在开发 C++中会变得十分方便与安全。栈内存的特性是让放入的数据得到自动的释放，并且依据先进后出的方式，以这种方式存放数据，便会使内存非常的整洁。而正因为栈区有这种先进后出的方式，同时系统会自动的释放出内存，这便使程序在运行占用内存后不会因为多次的人工释放的方式而导致产生许多的内存碎片。可以说，栈区在 C++中，虽然不那么灵活，但却是必不可少的部分。

3.2 C++堆内存

在 C/C++语言中，开发程序用途最广泛的工具就是指针。指针的用途相当广泛，其使用方法更是相当灵活，而要使用指针，便不像普通的变量那也简单。指针有自己的特性，即如果要使用，必须得为他在堆内存开辟空间，这种方式在类中非常适用。堆区并不像栈区一样，堆内存需要由程序员自己来使用语句开辟空间，然后，系统才将为程序提供堆区的内存空间，按照向高地址生长的方式分配不连续的内存空间。[2]这么一个过程，看似不难，但却会出现一个致命的错误。堆内存需要人为申请，在程序运行完之后，这片内存也必须人为的进行释放，而不是像栈区由系统自动释放。这样，在使用这片内存开发程序时就得有额外的释放的操作。很多初学者在申请内存后，却不进行释放，便随着程序运行，内存消耗越来越多，最终导致程序的崩溃。堆内存的申请和使用，看似简单，却也有很大的风险，开发程序时必须时刻关注此片内存的管理。

4.C++内存管理

4.1 new 与 delete 使用

在使用类中指针时，需要人来为指针开辟空间，以此才能使用。而开辟次空间，并不会像普通的变量一样，直接声明就可以进行使用，而是需要有特定的语法

——new。语句 new 允许操作人能够向系统申请需要用来存储的内存，而且类型不仅包括普通的 int 型，甚至还可以支持类等类型。在使用 new 时，系统也并不是仅仅只完成给程序分配内存的功能，系统还会对内存进行一个选择，然后将能够存储的空间分配给程序。分配的空间位于堆内存上，自然，在使用 new 之后，堆内存并不会自动地收回这个区域，而是需要程序员自己来将这片空间释放，所使用的方法则是 delete。Delete 在使用之后，会进行将程序使用的内存释放的工作。这项工作在使用堆内存的程序里是非常必要的，如果内存一直得不到释放，内存将会被占据的越来越多，这也是有一些程序越运行越慢的原因，最后则会导致程序崩溃。

new 与 delete 是运算符，不是函数，不仅可以实现内置数据类型(如 int、char) 的对象分配，也可以实现非内置数据类型(如 struct、class、enum 等)动态对象的需求。[3]通常，在针对为数组而申请存储空间的情况，new 与 delete 的用法又会不同。new[]和 delete[]会用来为数组申请空间来存储数据。两者之间，看着非常的相似，但功能上又有不同。new 只会申请出空间来存储，而因为 new[]申请的空间需要来存数组，所以 new[]还会额外申请一片空间来进行存储数组的属性。若使用 new 或 new[ ]申请了一块内存空间，在对象生命期结束时并没有及时进行内存的释放，则程序运行时间越长,占用内存就越多，可利用的内存 就越来越少，最终用尽全部内存，导致整个系统崩溃。[4]当然，delete 和 delete[]也不相同。但是 new[]和 delete 却不能交叉使用，比如 new[]已经申请额外内存，delete 却只释放一片内存，便同样会出现内存越存越多的情况。所以，new 与 delete 必须严格互相对照着使用，以防内存上不可挽回的错误。

4.2 内存泄漏与内存碎片

在开发一个程序中，new 与 delete 会被重复多次的使用，这也应该是借用堆内存来存储数据好的方式。但是，堆内存始终不能够如栈内存一样，由系统来进行内存的分配和释放，而在这重复的 new 与 delete 中仍然会产生一些失误，使得 new 与 delete 不能完美对照，但程序第一时间不会出现问题，这样的情况称为内存泄漏。内存泄漏虽然不会直接导致程序的崩溃，但正因为此，内存泄漏却又难以检查到。在大型程序中，代码量相当的大，而这一细小的错误如果开发时未注意到,很可能导致程序最终不能运行。此类错误，需要尽早排查，才能使程序更加健壮。

在使用内存申请算法时，也不可能保证十全十美，而使得一些细小的空闲的内存空间无法得到使用，同时再继续进行大于此部分内存的申请，会使这一细小的内存一直空闲，便变成了内存碎片。越界的内存可能保存其他变量的值，访问该内存变量的值，可能产生异常，导致程序的终止甚至崩溃。[5]内存碎片不同于内存泄漏的产生，但同样会使程序的运行效率降低，在 C++中依然是尽量去避免的。如果要使用内存，就必须管理好内存，否则便会出现适得其反的效果。

5. 结论

内存的管理对一个 C++程序来说十分的重要，C++的开发效率也因为内存变得比其他语言更低。C++不同于 JAVA 等语言，在开发时，在内存的管理上投入更多的精力是非常必要的。而且，如果管理好内存，对于开发 C++程序来说，更可谓是如虎添翼。虽然内存管理对于初学者来说并不友好，但只要在每次开发时，更加严谨，严格按 new 与 delete 或是其他规定写好代码，那么内存管理的基本工作也就已经完成了，距离开发更高效的 C++程序便可更进一步。

参考文献

[1] 韩雨涝.C 语言动态内存分配研究及应用[J].计算机时代,2009.5: 33-34

[2] 王文龙 C/C++ 数据内存分配和指针使用中若干问题的分析[A].喀什师范学院学报,2013(34):6

[3] 张 会 C++ 语言内存分配研究［J］.计算机时代，2014 (5) : 44-46

[4] 叶 煜，任华，雷静 C++ 下垃圾回收的分析[J].福建电 脑，2017，33(7) : 141- 142.

[5] Eri R.Hanly.C 语言详解[M].人民邮电出版社,2007

作者简介

第一作者：马可心（1997-），男，汉，四川省成都市，本科，四川大学锦城学院，研究方向：软件工程。

第二作者（通讯作者）：鲍正德（1989-），男，汉，黑龙江哈尔滨，研究生，四川大学锦城学院，研究方向：电子商务。

第三作者：唐娅雯（1999-），女，汉，四川省资阳市，本科，四川大学锦城学院，研究方向：信息管理、J2EE