at问题

C++中的内存管理是怎样的？

内存管理属于C++底层的知识，基本上是自己的知识盲区。看了几篇博客的总结，对内存管理和内存分配的分类似乎没有一个比较统一的说法，不过有些类似。这里主要参考两篇博客列举两种内存管理总结方式及其对应的内存分类方式，具体怎么回答就见仁见智啦，个人比较推荐第二种解读方式，感觉相对容易理解和记忆。

方式一： 参考自牛客网的 C++工程师面经汇总

内存管理

在C++中，虚拟内存分为代码段、数据段、BSS段、堆区、文件映射区以及栈区六部分。

代码段：包括只读存储区和文本区，其中**只读存储区存储字符串常量，文本区存储程序的机器代码**。

数据段：存储程序中**已初始化的全局变量和静态变量**。

BSS 段：存储**未初始化的全局变量和静态变量**（局部+全局），以及**所有被初始化为0的全局变量和静态变量**。

> 补充：BSS段的BSS是英文 Block Started by Symbol 的简写

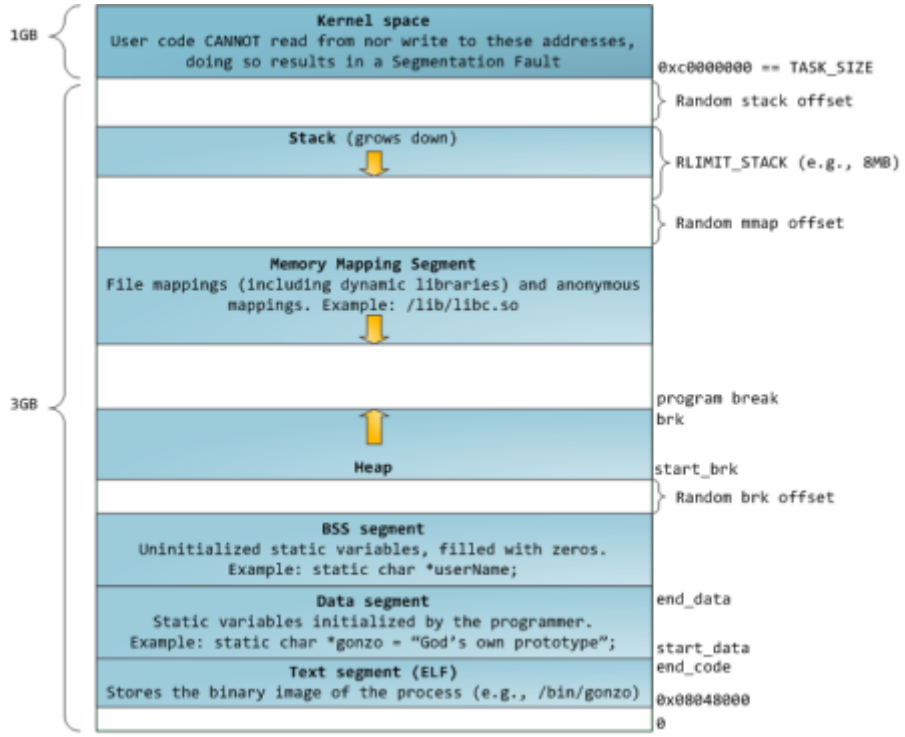
堆区：调用new/malloc函数时在堆区动态分配内存，同时需要调用delete/free来手动释放申请的内存。

栈区：使用栈空间存储**函数的返回地址、参数、局部变量、返回值**。

映射区：存储**动态链接库以及调用mmap函数进行的文件映射**。

内存分配

32bitCPU可寻址4G线性空间，每个进程都有各自独立的4G逻辑地址，其中0~3G是用户态空间，3~4G是内核空间，不同进程相同的逻辑地址会映射到不同的物理地址中。其逻辑地址其划分如下：



各段的说明如下：

区域	分段	内容
静态	**代码段(text segment)**	包括只读存储区和文本区，其中只读存储区存储字符串常量，文本区存储程序的机器代码
静态	**数据段(data segment)**	存储程序中已初始化的全局变量和静态变量
静态	**bss segment**	存储未初始化的全局变量和静态变量（局部+全局），以及所有被初始化为0的全局变量和静态变量，对于未初始化的全局变量和静态变量，程序运行main之前时会统一清零，即未初始化的全局变量编译器会初始化为0
动态	**堆(heap)**	使用栈空间存储函数的返回地址、参数、局部变量、返回值，从高地址向低地址增长。在创建进程时会有一个最大栈大小，Linux可以通过ulimit命令指定
动态	**栈 (stack)**	使用栈空间存储函数的返回地址、参数、局部变量、返回值，从高地址向低地址增长。在创建进程时会有一个最大栈大小，Linux可以通过ulimit命令指定
动态	**映射区(memory mapping segment)**	存储动态链接库等文件映射、申请大内存（malloc时调用mmap函数）

方式二：参考自这篇博客，原文链接：https://blog.csdn.net/cherrydreamsover/article/details/81627855

内存管理

栈区（stack）：由编译器自动分配与释放，存放为运行时函数分配的局部变量、函数参数、返回值、返回地址等。

堆区（heap）：一般由程序员自动分配（使用new分配），如果程序员没有释放（使用delete释放），程序结束时系统自动回收。

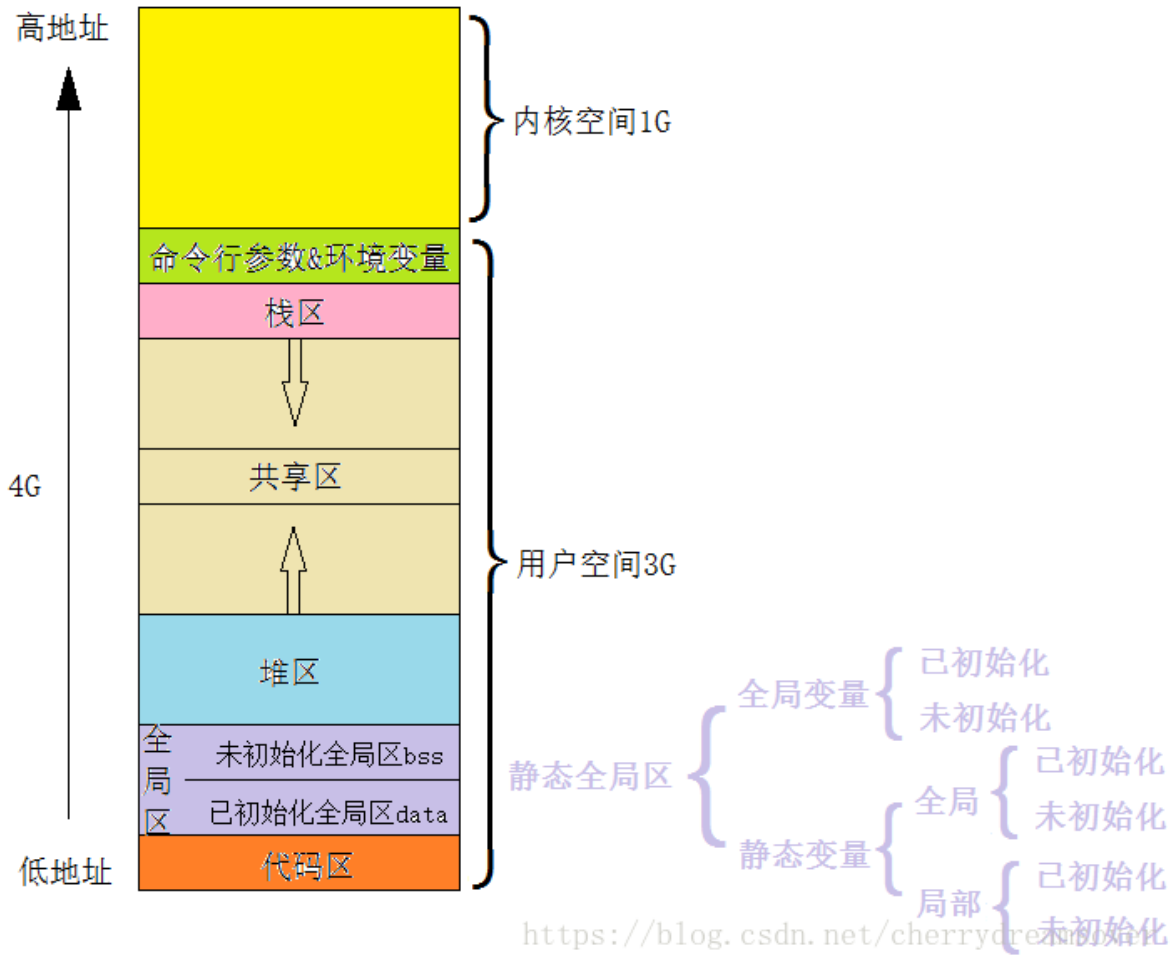
全局区（静态区）：存放全局变量、静态数据、常量。程序结束后由系统释放。全局区分为已初始化全局区（data）和未初始化全局区（bss）。

文字常量区：存放常量字符串，程序结束后由系统释放。

代码区：存放函数体（类成员函数和全局区）的二进制代码。

内存分配

先看下面这张图：



具体请看：

分配方式	内容
从静态存储区分配	内存在程序编译的时候已经分配好，这块内存在程序的整个运行期间都存在。例如全局变量，static变量。
在栈上创建	●在执行函数时，函数内局部变量的存储单元可以在栈上创建，函数执行结束时，这些内存单元会自动被释放。 ●栈内存分配运算内置于处理器的指令集中，效率高，但是分配的内存容量有限。
从堆上分配（也称为**动态内存分配**）	● 程序在运行的时候使用malloc或者new申请任意多少的内存，程序员自己负责在何时用free或delete释放内存。 ● 动态内存的生命周期由程序员决定，使用非常灵活，但如果在堆上分配了空间，既有责任回收它，否则运行的程序会出现内存泄漏，频繁的分配和释放不同大小的堆空间将会产生内存碎片。

拓展

上面第二位博主总结的太好了，顺便摘抄其剩下的总结。

堆和栈的区别

● **管理方式不同：**栈是由编译器自动申请和释放空间，堆是需要程序员手动申请和释放；

● **空间大小不同：**栈的空间是有限的，在32位平台下，VC6下默认为1M，堆最大可以到4G；

● **能否产生碎片：**栈和数据结构中的栈原理相同，在弹出一个元素之前，上一个已经弹出了，不会产生碎片，如果不停地调用malloc、free对造成内存碎片很多；

● **生长方向不同：**堆生长方向是向上的，也就是向着内存地址增加的方向，栈刚好相反，向着内存减小的方向生长。

● **分配方式不同：**堆都是动态分配的，没有静态分配的堆。栈有静态分配和动态分配。静态分配是编译器完成的，比如局部变量的分配。动态分配由 malloc 函数进行分配，但是栈的动态分配和堆是不同的，它的动态分配是由编译器进行释放，无需我们手工实现。

● **分配效率不同：**栈的效率比堆高很多。栈是机器系统提供的数据结构，计算机在底层提供栈的支持，分配专门的寄存器来存放栈的地址，压栈出栈都有相应的指令，因此比较快。堆是由库函数提供的，机制很复杂，库函数会按照一定的算法进行搜索内存，因此比较慢。

静态全局变量与全局变量的区别

● 静态全局变量和全局变量都属于 **常量区**。

● 静态全局区只在本文件中有效，别的文件想调用该变量是调不了的，而全局变量可以在别的文件中调用。

● 如果别的文件中定义了一个该全局变量相同的变量名，是会出错的。

静态局部变量和局部变量的区别

● 静态局部变量是属于常量区的，而函数内部的局部变量属于栈区。

● 静态局部变量在该函数调用结束时，不会销毁，而是随整个程序结束而结束，但是别的函数调用不了该变量，局部变量随该函数的结束而结束。

● 如果定义这两个变量的时候没有初始值时，静态局部变量会自动定义为0，而局部变量就是一个随机值。

● 静态局部变量在编译期间只赋值一次，以后每次函数调用时，不再赋值，调用上次的函数调用结束时的值。局部变量在调用期间，每调用一次，赋一次值。

补充

对于内存管理，有一个很重要的概念就是动态内存管理，通过malloc/new、free/delete来实现，具体可以参考之前总结的C++基础第 11、12个问题的内容，也可以深入阅读参考资料中最后一篇文章。

参考资料

请你说一说C++的内存管理是怎样的？

C/C++程序内存的分配

C语言中内存分配

C/C++动态内存管理malloc/new、free/delete的异同