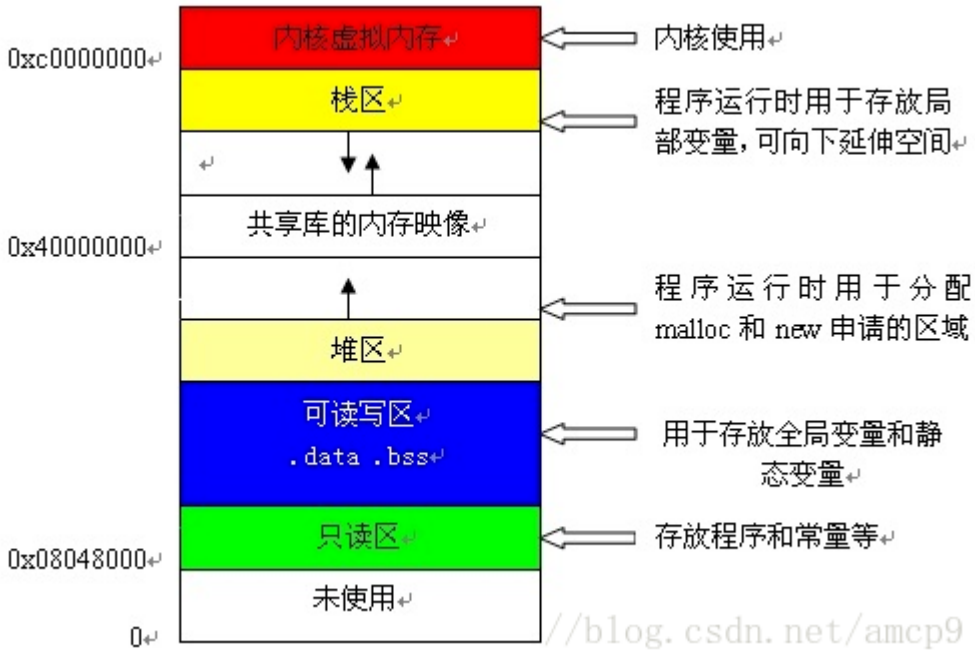
操作系统堆与栈

当一个程序运行时，其**RAM**存储方式是按照一定的区域划分的，以**C**为例



内存中的栈区处于相对较高的地址向较低的地址拓展，由操作系统决定的最高地址，所以它是一块连续的内存空间。

栈中分配局部变量空间，堆区是低地址向高地址拓展，用于分配程序员申请的内存空间。另外还有静态区是分配静态变量，全局变量空间的；只读区是分配常量和程序代码空间的；以及其他一些分区。

栈：

栈是为执行线程留出的内存空间。当函数被调用的时候，栈顶为局部变量和一些 bookkeeping 数据预留块。当函数执行完毕，块就没有用了，可能在下次的函数调用的时候再被使用。栈通常用后进先出（LIFO）的方式预留空间；因此最近的保留块（reserved block）通常最先被释放。这么做可以使跟踪堆栈变的简单；从栈中释放块（free block）只不过是指针的偏移而已。

1.堆包含一个链表来维护已用和空闲的内存块。在堆上新分配（用 new 或者 malloc）内存是从空闲的内存块中找到一些满足要求的合适块。这个操作会更新堆中的块链表。这些元信息也存储在堆上，经常在每个块的头部一个很小区域。

2.堆的增加新快通常从地地址向高地址扩展。因此你可以认为堆随着内存分配而不断的增加大小。如果申请的内存大小很小的话，通常从底层操作系统中得到比申请大小要多的内存。

3.申请和释放许多小的块可能会产生如下状态：在已用块之间存在很多小的空闲块。进而申请大块内存失败，虽然空闲块的总和足够，但是空闲的小块是零散的，不能满足申请的大小，。这叫做“堆碎片”。当旁边有空闲块的已用块被释放时，新的空闲块可能会与相邻的空闲块合并为一个大的空闲块，这样可以有效的减少“堆碎片”的产生。

堆：

堆（heap）是为动态分配预留的内存空间。和栈不一样，从堆上分配和重新分配块没有固定模式；你可以在任何时候分配和释放它。这样使得跟踪哪部分堆已经被分配和被释放变的异常复杂；有许多定制的堆分配策略用来为不同的使用模式下调整堆的性能。

堆和栈是两种内存分配的两个统称。可能有很多种不同的实现方式，但是实现要符合几个基本的概念:

1.对栈而言，栈中的新加数据项放在其他数据的顶部，移除时你也只能移除最顶部的数据（不能越位获取）。

2.对堆而言，数据项位置没有固定的顺序。你可以以任何顺序插入和删除，因为他们没有“顶部”数据这一概念。

堆和栈是一个统称，可以有很多的实现方式。计算机程序通常有一个栈叫调用栈，用来存储当前函数调用相关的信息（比如：主调函数的地址，局部变量），因为函数调用之后需要返回给主调函数，也就是说，一旦函数调用返回，局部变量将释放

栈附属于线程，因此当线程结束时栈被回收。堆通常通过运行时在应用程序启动时被分配，当应用程序（进程）退出时被回收。 。当线程被创建的时候，设置栈的大小。在应用程序启动的时候，设置堆的大小，但是可以在需要的时候扩展（分配器向操作系统申请更多的内存）

栈和堆都是用来从底层操作系统中获取内存的。

在多线程环境下每一个线程都可以有他自己完全的独立的栈，但是他们共享堆。并行存取被堆控制而不是栈。

对于C#而言：

托管堆上部署了所有引用类型。这很容易理解。当创建一个应用类型变量时：

object reference = new object();

关键字new将在托管堆上分配内存空间，并返回一个该内存空间的地址。左边的reference位于栈上，是一个引用，存储着一个内存地址；而这个地址指向的内存（位于托管堆）里存储着其内容（一个System.Object的实例）。下面为了方便，简称引用类型部署在托管推上

栈与堆的主要区别：

申请方式与回收方式的区别：

栈：栈 是系统自动分配空间的，例如我们定义一个 char a；系统会自动在栈上为其开辟空间。而堆 则是程序员根据需要自己申请的空间，例如malloc或者new。由于栈上的空间是自动分配自动回收的，所以栈上的数据的生存周期只是在函数的运行过程中，运行后就释放掉，不可以再访问。

堆：而堆上的数据只要程序员不释放空间，就一直可以访问到，不过缺点是一旦忘记释放会造成内存泄露。

管理方式的区别：

栈：栈编译器自动管理，无需程序员手工控制

堆：堆空间的申请释放工作由程序员控制，容易产生内存泄漏。

空间连续性的区别：

栈：栈是向低地址扩展的数据结构，是一块连续的内存区域。这句话的意思是栈顶的地址和栈的最大容量是系统预先规定好的，当申请的空间超过栈的剩余空间时，将提示溢出。因此，用户能从栈获得的空间较小。

堆：堆是向高地址扩展的数据结构，是不连续的内存区域。因为系统是用链表来存储空闲内存地址的，且链表的遍历方向是由低地址向高地址。由此可见，堆获得的空间较灵活，也较大。

地址的增长方向的区别：

栈：栈的增长方向是向着内存地址减小的方向。

堆：堆的增长方向是向着内存地址增加的方向；

是否产生碎片的区别：

栈：对于栈来讲，不会存在这个问题。

堆：对于堆来讲，频繁的malloc/free（new/delete）势必会造成内存空间的不连续，从而造成大量的碎片，使程序效率降低（虽然程序在退出后操作系统会对内存进行回收管理）。

分配效率的区别：

栈：只要栈的剩余空间大于所申请空间，系统将为程序提供内存，否则将报异常提示栈溢出。

堆：在堆内存中搜索可用的足够大的空间，如果没有足够大的空间（可能是由于内存碎片太多），就有需要操作系统来重新整理内存空间，这样就有机会分到足够大小的内存，然后返回。显然，堆的效率比栈要低得多

（同时这也是C#中的装箱与拆箱损耗性能的原因：装箱就是把栈上的数据放到堆上,拆箱就是把堆上的数据挪到栈上.这里多了数据的转换...需要申请内存等操作）

存储内容的区别：

值类型的实例通常是在线程栈上分配的（静态分配），但是在某些情形下可以存储在堆中。

（同时包含在函数调用时，主函数中函数调用后的下一条指令（函数调用语句的下一条可执行语句）的地址，函数的各个参数，函数中的局部变量。注意静态变量是不入栈的。 当本次函数调用结束后，按照先进后入的方式直到主函数中的下一条指令，程序由该点继续运行。 ）

引用类型的对象总是在进程堆中分配（动态分配）。