堆区与栈区

内存分区

> - 堆区
> - 栈区
> - 静态区
> - 常量区
> - 代码区

1、堆区

- 【1】**就是通过`new`、`malloc`、`realloc`分配的内存块，编译器不会负责它们的释放工作，需要用程序区释放`delete`。分配方式类似于数据结构中的链表。“内存泄漏”通常说的就是堆区。**
- 【2】**一般由程序员分配释放，若是程序员不释放，程序结束时可能由操作系统回收，类似于链表，在内存中的分布是不连续的，它们是不同区域的内存块通过指针链接起来的。**一旦某一节点从链中断开，我们要人为的把所有断开的节点从内存中释放。

2、栈区

- 【1】**存放函数的参数值、局部变量等，由编译器自动分配和释放**，通常在函数执行完后就释放了，**其操作方式类似于数据结构中的栈。**栈内存分配运算内置于CPU的指令集，效率很高，但是分配的内存量有限。
- 【2】**由编译器自动分配释放，存放函数的参数值，局部变量的值等，内存的分配是连续的，类似于平时我们说的栈，**它的内存分配是连续分配的，即，所分配的内存是在一块连续的内存区域内．**当我们声明变量时，那么编译器会自动接着当前栈区的结尾来分配内存。**
- 【3】**定义的局部变量按照先后定义的顺序依次压入栈中，也就是说相邻变量的地址之间不会存在其它变量。栈的内存地址生长方向与堆相反，由高到底，所以后定义的变量地址低于先定义的变量，**比如上面代码中变量 s 的地址小于变量 b 的地址，p2 地址小于 s 的地址。**栈中存储的数据的生命周期随着函数的执行完成而结束。**

3、静态区

- 【1】**全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量是在一块区域，未初始化的全局全局变量和未初始化的静态变量在相邻的另一块区域。程序结束后由系统释放。**
- 【2】**全局变量使用`static`修饰，**全局静态区的空间大小和堆的大小差不多。

4、常量区

- **常量存储在这里，不允许被修改**

5、代码区

- **存放代码**