# C语言生命周期与作用域

——黄昌彬（2018.11.8）

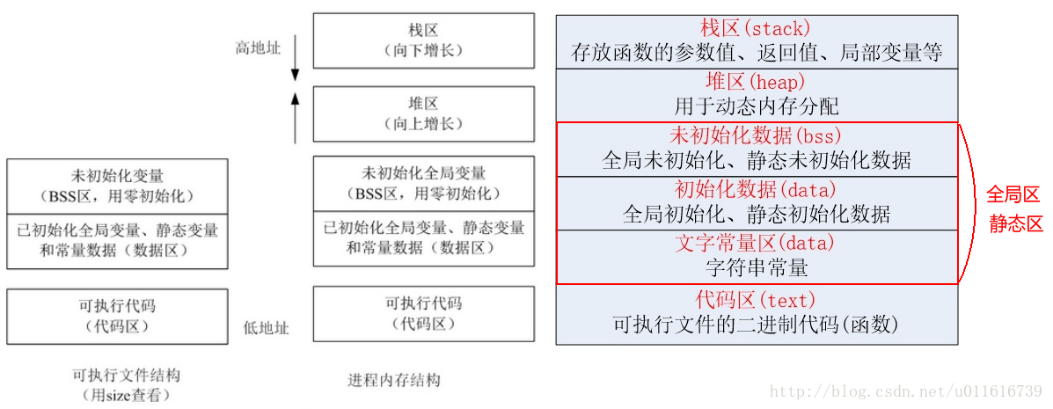
## C语言的内存管理

C源代码经过预处理、编译、汇编与链接成为一个可执行程序。

C程序在运行之前，未加载到内存前，可执行程序内部已分为3部分信息，BSS区（未初始化数据区）、Data（全局初始化数据区）、Text(代码区)。

运行可执行程序，程序被加载到内存中，除了根据已有信息分配这三个区外，还额外增加了两个区，栈区（stack）和堆（heap）区。

具体关系如图所示。



注：

1. 所有未初始化的全局变量与静态变量，编译器会默认赋初值为0；
2. 程序在加载到内存前，代码区和全局区（data与bss）的大小固定，且程序运行期间不

能改变；

1. data区与bss区中数据的生命周期为整个程序运行期间；
2. data区、bss区与text区是由编译器在编译时分配的，堆和栈实在程序运行时分配的。

## 变量生命周期与作用域

作用域分为全局作用域、局部作用域与文件作用域。

1. **普通全局变量**

作用域：全局作用域（只在一个源文件定义，多个源文件均可以使用，外部引用需用

extern关键字声明）

生命周期：从编译到程序运行期间一直存在

内存分布：data区（初始化）与bss区（未初始化）

1. **静态全局变量**

作用域：文件作用域（只能在定义的文件中使用，不允许外部调用）

生命周期：从编译到程序运行期间一直存在

内存分布：data区（初始化）与bss区（未初始化）

1. **普通局部变量**

作用域：局部作用域

生命周期：从定义到程序出局部作用域即被销毁（即属于某个{}，程序执行到{即分配内

存，执行到}即销毁，下次进入再重新分配。）

内存分布：栈区

1. **静态局部变量**

作用域：局部作用域

生命周期：从编译到程序运行期间一直存在(编译阶段即分配内存，重新进入也不会重新分配)

内存分布：data区（初始化）与bss区（未初始化）

1. **寄存器变量**

作用域：局部作用域

生命周期：无

内存分布：使用CPU的寄存器，无地址，不占用内存。

注：

为了提高效率，C 语言允许将局部变量的值放在 CPU 中的寄存器中，这种变量叫“寄存器变量”，用关键字register作声明。

只有局部自动变量和形式参数可以作为寄存器变量；

一个计算机系统中的寄存器数目有限，不能定义任意多个寄存器变量；

局部静态变量不能定义为寄存器变量。

**总结：**

全局变量和静态变量是在程序编译期就分配到data段或bss段中的，在整个程序的运行期，数据段和bss段中的内容是不会发生改变的。所以所有的全局变量和静态变量的生命周期必定是从程序编译到程序运行结束。而且如果全局变量和静态变量没有初始化，会统一默认初始化为0.

而局部变量是在程序的运行时，在栈中分配的内存，所以生命周期一定是从程序运行的某一个时刻开始。

**关于static变量使用的一些tips:**

1. 若全局变量仅在单个C文件中访问，则可以将这个变量修改为静态全局变量，以降低模

块间的耦合度；

1. 若全局变量仅由单个函数访问，则可以将这个变量改为该函数的静态局部变量，以降低

模块间的耦合度；

1. 设计和使用访问动态全局变量、静态全局变量、静态局部变量的函数时，需要考虑重入

问题，因为他们都放在静态数据存储区，全局可见；

1. 如果我们需要一个可重入的函数，那么，我们一定要避免函数中使用static变量(这样

的函数被称为：带“内部存储器”功能的的函数)；

1. 函数中必须要使用static变量情况:比如当某函数的返回值为指针类型时，则必须是

static的局部变量的地址作为返回值，若为auto类型，则返回为错指针。