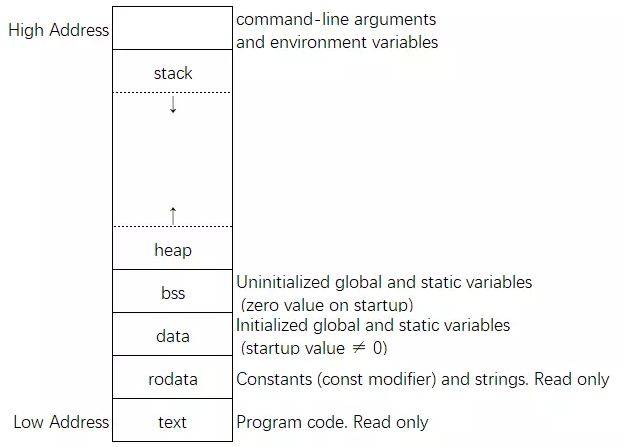
**C语言内存分配**

**1、内存分配图**



**2、各段内存分配**

bss段：

BSS（Block Started by Symbol）属于静态内存分配，通常是指用来存放程序中未初始化的全局变量和静态变量的一块内存区域。特点是可读写。

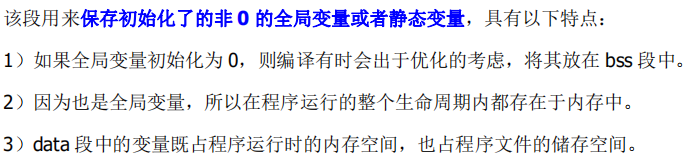
BSS段在磁盘上不是真的占用变量大小的空间，它仅仅记录了变量所需要的大小（占位符），也就是说，在该段中记录了所有未初始化全局变量与局部静态变量的大小总和，至于每个变量的大小则存储在符号表的size属性中。当可执行文件加载运行前，会为BSS段中的变量分配足够的空间并全部自动清零（因此，才有未初始化的全局变量的值为0的说法）。

BSS段主要是为了节省可执行文件在磁盘上所占的空间，对未初始化的大型数组的节省效率比较明显。默认情况下，编译器会把初始化值为0的变量（比如static int a = 0）放在BSS段。但可以通过“#pragma explicit\_zero\_data on”把初始值为零的变量放到data段，而不是bss段。

data段：

　　数据段（data segment）通常是指用来存放程序中已初始化的全局变量的一块内存区域。

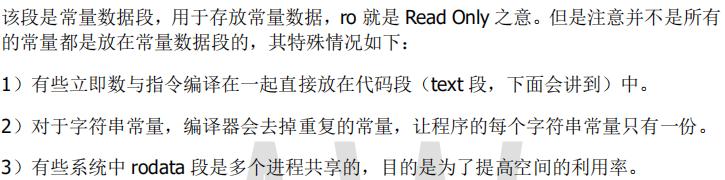
数据段属于静态内存分配。



rodata段

只读数据段（read only data segment）

用于存储使用const修饰的变量，或者只读类型的字符串等。



text段：

　　代码段（code segment/text segment）通常是指用来存放程序执行代码的一块内存区域。

　　这部分区域的大小在程序运行前就已经确定，并且内存区域通常属于只读(某些架构也允许代码段为可写，即允许修改程序)。

在代码段中，也有可能包含一些只读的常数变量，例如字符串常量等。



堆（heap）：

　　堆是用于存放进程运行中被动态分配的内存段，它的大小并不固定，可动态扩张或缩减。

　　当进程调用malloc等函数分配内存时，新分配的内存就被动态添加到堆上（堆被扩张）；

　　当利用free等函数释放内存时，被释放的内存从堆中被剔除（堆被缩减）。

栈(stack)：

　　 栈又称堆栈，是用户存放程序临时创建的局部变量，

　　也就是说我们函数括弧“{}”中定义的变量（但不包括static声明的变量，static意味着在数据段中存放变量）。

　　除此以外，在函数被调用时，其参数也会被压入发起调用的进程栈中，并且待到调用结束后，函数的返回值也会被存放回栈中。

　　栈特别方便用来保存/恢复调用现场。

　　从这个意义上讲，我们可以把堆栈看成一个寄存、交换临时数据的内存区。

**3、总结**

1. 一个程序本质上都是由 bss段、data段、text段三个组成的。

这样的概念，不知道最初来源于哪里的规定，但在当前的计算机程序设计中是很重要的一个基本概念。而且在嵌入式系统的设计中也非常重要，牵涉到嵌入式系统运行时的内存大小分配，存储单元占用空间大小的问题。

1. 在采用段式内存管理的架构中（比如intel的80x86系统），bss段通常是指用来存放程序中未初始化的全局变量的一块内存区域，一般在初始化时bss 段部分将会清零。bss段属于静态内存分配，即程序一开始就将其清零了。

比如，在C语言之类的程序编译完成之后，已初始化的全局变量保存在.data 段中，未初始化的全局变量保存在.bss 段中。

1. text和data段都在可执行文件中（在嵌入式系统里一般是固化在镜像文件中），由系统从可执行文件中加载；而bss段不在可执行文件中，由系统初始化。