一个程序本质上都是由 BSS 段、data段、text段三个组成的。这种概念在当前的计算机程序设计中是非常重要的一个基本概念，并且在嵌入式系统的设计中也非常重要，牵涉到嵌入式系统执行时的内存大小分配，存储单元占用空间大小的问题。

* BSS段：在採用段式内存管理的架构中。BSS段（bss segment）一般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。

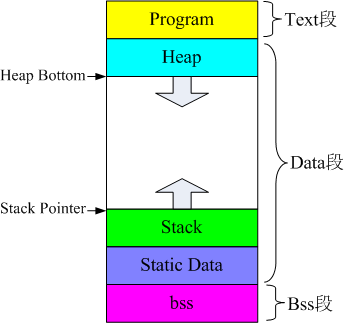
BSS是英文Block Started by Symbol的简称。

BSS段属于**静态内存分配**。

* 数据段：在採用段式内存管理的架构中，数据段（data segment）一般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。
* 代码段：在採用段式内存管理的架构中，代码段（text segment）一般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执行前就已经确定，而且内存区域属于仅仅读。

在代码段中。也有可能包括一些仅仅读的常数变量，比如字符串常量等。

程序[编译](http://www.nowamagic.net/librarys/veda/tag/%E7%BC%96%E8%AF%91" \t "https://www.cnblogs.com/mfmdaoyou/p/_blank)后生成的目标文件至少含有这三个段。这三个段的大致结构图例如以下所看到的：



当中.text即为代码段，为仅仅读。.bss段包括程序中未初始化的全局变量和static变量。

data段包括三个部分：heap(堆)、stack(栈)和静态数据区。

* 堆（heap）：堆是用于存放进程执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加入到堆上（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， 是用户存放程序暂时创建的局部变量，也就是说我们函数括弧“{}”中定义的变量（但不包含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。其參数也会被压入发起调用的进程栈中。而且待到调用结束后。函数的返回值也会被存放回栈中。

因为栈的先进先出特点，所以栈特别方便用来保存/恢复调用现场。从这个意义上讲，我们能够把堆栈看成一个寄存、交换暂时数据的[内存](http://www.nowamagic.net/librarys/veda/tag/%E5%86%85%E5%AD%98" \t "https://www.cnblogs.com/mfmdaoyou/p/_blank)区。

当程序在运行时动态分配空间（C中的malloc函数），所分配的空间就属于heap。其概念与数据结构中“堆”的概念不同。

stack段存放函数内部的变量、參数和返回地址，其在函数被调用时自己主动分配。訪问方式就是标准栈中的LIFO方式。

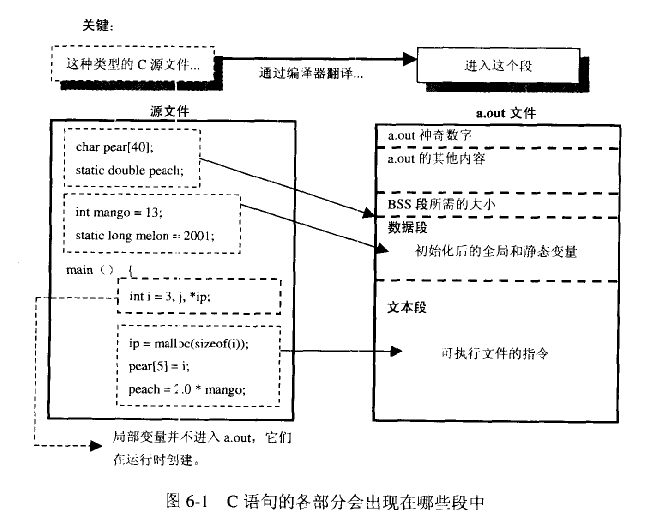
（由于函数的局部变量存放在此，因此其訪问方式应该是栈指针加偏移的方式，否则若通过push、pop操作来訪问相当麻烦）

data段中的静态数据区存放的是程序中已初始化的全局变量、静态变量和常量。

在採用段式内存管理的架构中（比方intel的80x86系统），BSS 段（Block Started by Symbol segment）一般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。即程序一開始就将其清零了。

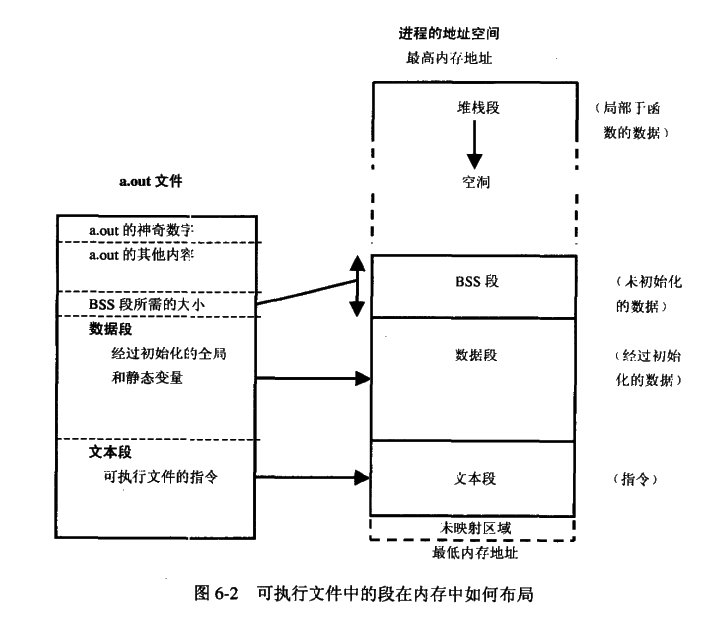
比方，在C语言之类的程序编译完毕之后，已初始化的全局变量保存在.data 段中，未初始化的全局变量保存在.bss 段中。

text和data段都在可运行文件里（在嵌入式系统里通常是固化在镜像文件里）。由系统从可运行文件里载入；而BSS段不在可运行文件里，由系统初始化。



图引自《C专家编程》

BSS段仅仅保存没有值的变量。所以其实它并不须要保存这些变量的映像。执行时所须要的BSS段大小记录在目标文件里，但BSS段并不占领目标文件的不论什么空间。



//main.c

int a = 0; //全局初始化区

char \*p1; //全局未初始化区

main()

{

static int c =0。 //全局（静态）初始化区

int b; //栈

char s[] = "abc"; //栈

char \*p2; //栈

char \*p3 = "123456"; //"123456\0"在常量区，p3在栈上。

p1 = (char \*)malloc(10);

p2 = (char \*)malloc(20); //分配得来得10和20字节的区域就在堆区。

}