C语言编程-----程序的内存布局

一：C语言程序的存储区域

由C语言代码（文本文件）形成可执行程序（二进制文件），需要经过编译-汇编-连接三个阶段。编译过程把C语言文本文件生成汇编程序，汇编过程把汇编程序形成二进制机器代码，连接过程则将各个源文件生成的二进制机器代码文件组合成一个文件。

C语言编写的程序经过编译-连接后，将形成一个统一文件，它由几个部分组成。在程序运行时又会产生其他几个部分，各个部分代表了不同的存储区域：

1.代码段（Code或Text）

代码段由程序中执行的机器代码组成。在C语言中，程序语句进行编译后，形成机器代码。在执行程序的过程中，CPU的程序计数器指向代码段的每一条机器代码，并由处理器依次运行。

2.只读数据段（RO data）

只读数据段是程序使用的一些不会被更改的数据，使用这些数据的方式类似直表式的操作，由于这些变量不需要更改，因此只需要放置在只读存储器中即可。

3.已初始化读写数据段（RW data）

已初始化数据是在程序中声明，并且具有初值的变量，这些变量需要占用存储器的空间，在程序执行时它们需要位于可读写的内存区域内，并具有初值，以供程序运行时读写。

4.未初始化数据段（BSS）

未初始化数据是在程序中声明，但是没有初始化的变量，这些变量在程序运行之前不需要占用存储器的空间。

5.堆（heap）

堆内存只在程序运行时出现，一般由程序员分配和释放。在具有操作系统的情况下，如果程序没有释放，操作系统可能在程序（例如一个进程）结束后回收内存。

6.栈（stack）

栈内存只在程序运行时出现，在函数内部使用的变量、函数的参数以及返回值将使用栈空间，栈空间由编译器自动分配和释放。

看一个例子：

```
int a = 0; //全局初始化区。        data段
 static int b=20; //全局初始化区。  data段
char *p1; //全局未初始化区          bss段
const int A = 10; //               rodata段
void main(void)
{
int b; //栈
char s[] = "abc"; //栈
char *p2; //栈
static int c = 0; //全局（静态）初始化区 .data段
char *p3 = "123456"; //123456\0在常里区，p3 在栈上。
p1 = (char*) malloc(10); //分配得来的10和20个字节的区域就在堆区
p2 = (char*) malloc(20);
strcpy(p1, "123456"); //123456\0 在常里区，编译器可能会将它与p3所指向的"123456"优化成一个地方
}
```

代码段、只读数据段、读写数据段、未初始化数据段属于静态区域，而堆和栈属于动态区域。代码段、只读数据段和读写数据段将在链接之后产生，未初始化数据段将在程序初始化的时候开辟，而堆和栈将在程序的运行中分配和释放。C语言程序分为映像和运行时两种状态。在编译-连接后形成的映像中，将只包含代码段（Text）、只读数据段（RO Data）和读写数据段（RW Data）。在程序运行之前，将动态生成未初始化数据段（BSS），在程序的运行时还将动态形成堆（Heap）区域和栈（Stack）区域。一般来说，在静态的映像文件中，各个部分称之为节（Section），而在运行时的各个部分称之为段（Segment）。如果不详细区分，可以统称为段。

在C语言的程序中，对变量的使用还有以下几点需要注意：

1.函数体中定义的变量通常是在栈上，不需要在程序中进行管理，由编绎器处理。

2.用malloc,calloc,realloc等分配内存的函数所分配的内存空间在堆上，程序必须保证在使用free释放，否则会发生内存泄漏。

3.所有函数体外定义的是全局变量，加了static后的变量不管是在函数内部或外部都放在全局区。

4.使用const定义的变量将放于程序的只读数据区。

栈空间主要用于以下3数据的存储：

1.函数内部的动态变量

2.函数的参数

3.函数的返回值

栈空间是动态开辟与回收的。在函数调用过程中，如果函数调用的层次比较多，所需要的栈空间也逐渐加大，对于参数的传递和返回值，如果使用较大的结构体，在使用的栈空间也会比较大。