Static翻译出来是“静态”“静止”的意思，在C语言中的意思其实和它的本意差不多，表示“静态”或者“全局”的意思，用来修饰变量和函数。经static修饰过后的变量或者函数的作用域或者存储域会发生变化，

一、c程序的内存分布

既然static是用来修饰变量和函数的，而变量和函数又是组成c程序必不可少的，C程序的内存分布图如下。

C程序由下面5部分组成:

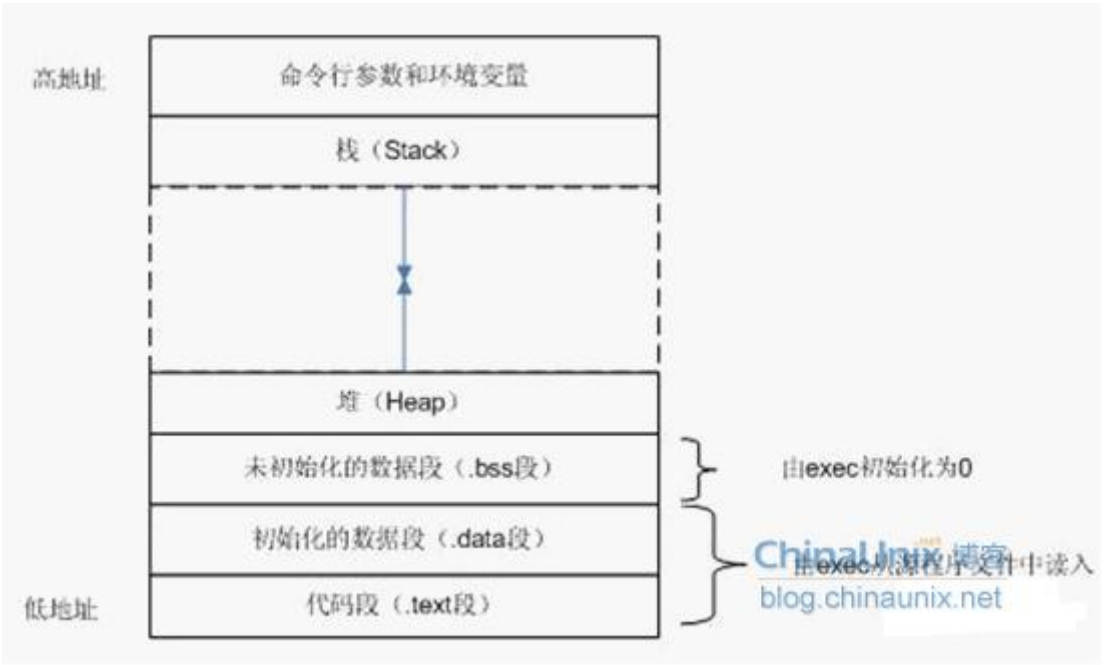
1）正文段——CPU执行的机器指令部分；一个程序只有一个副本；只读，防止程序由于意外事故而修改自身指令；

2）初始化数据段（数据段）——在程序中所有赋了初值的全局变量，存放在这里。

3）非初始化数据段（bss段）——在程序中没有初始化的全局变量；内核将此段初始化为0。

4）栈——增长方向：自顶向下增长；自动变量以及每次函数调用时所需要保存的信息（返回地址；环境信息）。

5）堆——动态存储区。是向高地址扩展的数据类型，是自下向上的扩展方式。



c程序内存分布图

上面的C程序分布图很明显的告诉我们，变量是存储在栈区或者堆区或者bss段或者data段，变量的存储域为什么会有所不同呢？其实原因很简单，说白了就是与他们定义在程序的不同地方，有没有static关键字修饰有关啦，定义在不同的地方也说明了他们有着不同的作用域。

二、static修饰的变量

1. 全局静态变量

在全局变量之前加上关键字static，全局变量就被定义成为一个全局静态变量。

1）内存中的位置：静态存储区（静态存储区在整个程序运行期间都存在）

2）初始化：未经初始化的全局静态变量会被程序自动初始化为0（自动对象的值是任意的，除非他被显示初始化）

3）作用域：全局静态变量在声明他的文件之外是不可见的。准确地讲从定义之处开始到文件结尾。

定义全局静态变量的好处：

<1>不会被其他文件所访问，修改

<2>其他文件中可以使用相同名字的变量，不会发生冲突。

2. 局部静态变量

在局部变量之前加上关键字static，局部变量就被定义成为一个局部静态变量。

1）内存中的位置：静态存储区

2）初始化：未经初始化的局部静态变量会被程序自动初始化为0（自动对象的值是任意的，除非他被显示初始化）

3）作用域：作用域仍为局部作用域，当定义它的函数或者语句块结束的时候，作用域随之结束。

注：当static用来修饰局部变量的时候，它就改变了局部变量的存储位置，从原来的栈中存放改为静态存储区。但是局部静态变量在离开作用域之后，并没有被销毁，而是仍然驻留在内存当中，直到程序结束，只不过我们不能再对他进行访问。

当static用来修饰全局变量的时候，它就改变了全局变量的作用域（在声明他的文件之外是不可见的），但是没有改变它的存放位置，还是在静态存储区中。

三、Static修饰的函数

在函数的返回类型前加上关键字static，函数就被定义成为静态函数。

函数的定义和声明默认情况下是extern的，但静态函数只是在声明他的文件中可见，不能被其他文件所用。

定义静态函数的好处：

<1> 其他文件中可以定义相同名字的函数，不会发生冲突

<2> 静态函数不能被其他文件所用。

存储说明符auto，register，extern，static，对应两种存储期：自动存储期和静态存储期。

auto和register对应自动存储期。具有自动存储期的变量在进入声明该变量的程序块时被建立，它在该程序块活动时存在，退出该程序块时撤销。

关键字extern和static用来说明具有静态存储期的变量和函数。用static声明的局部变量具有静态存储持续期（static storage duration），或静态范围（static extent）。虽然他的值在函数调用之间保持有效，但是其名字的可视性仍限制在其局部域内。静态局部对象在程序执行到该对象的声明处时被首次初始化。

四、总结

（1）第一个作用：隐藏。

当我们同时编译多个文件时，所有未加static前缀的全局变量和函数都具有全局可见性。为理解这句话，我举例来说明。我们要同时编译两个源文件，一个是a.c，另一个是main.c。

下面是a.c的内容

```
#include<cstdio>增加这条语句
char a = 'A'; // global variable
void msg()
{
    printf("Hello\n");
}
```

下面是 main.c 的内容

```
int main(void)
{
    extern char a;    // extern variable must be declared before use
    printf("%c ", a);
    (void)msg();
    return 0;
}
```

程序的运行结果是： A Hello

你可能会问：为什么在a.c中定义的全局变量a和函数msg能在main.c中使用？前面说过，所有未加static前缀的全局变量和函数都具有全局可见性，其它的源文件也能访问。此例中，a是全局变量，msg是函数，并且都没有加static前缀，因此对于另外的源文件main.c是可见的。

如果加了static，就会对其它源文件隐藏。例如在a和msg的定义前加上static，main.c就看不到它们了。利用这一特性可以在不同的文件中定义同名函数和同名变量，而不必担心命名冲突。Static可以用作函数和变量的前缀，对于函数来讲，static的作用仅限于隐藏，而对于变量，static还有下面两个作用。

（2）static的第二个作用是保持变量内容的持久。存储在静态数据区的变量会在程序刚开始运行时就完成初始化，也是唯一的一次初始化。共有两种变量存储在静态存储区：全局变量

和static变量，只不过和全局变量比起来，static可以控制变量的可见范围，说到底static还是用来隐藏的。

（3）static的第三个作用是默认初始化为0。其实全局变量也具备这一属性，因为全局变量也存储在静态数据区。在静态数据区，内存中所有的字节默认值都是0x00，某些时候这一特点可以减少程序员的工作量。

最后对static的三条作用做一句话总结。首先static的最主要功能是隐藏，其次因为static变量存放在静态存储区，所以它具备持久性和默认值0。

下面是main.c的内容

除了头文件，需要声明函数：void msg();

```
int main(void)
{
   extern char a;    // extern variable must be declared before use
    printf("%c ", a);
    (void)msg();
   return 0;
}
```

理解：

static 三个用法：

1：加在 全局变量

2：加在 局部变量

3：加在 函数

作用：

1 和 3 都是限制访问权限，只能本文件访问，

2 ，将 局部变量 放到 全局变量一个区域，但是访问权限只有 本函数

说明：

1：static 声明的变量 在编译的时候就分配好了内存，因此 不能初始化为不确定的值 例如：time 或者别的变量

2：全局变量也是如此