static变量及其作用，C语言static变量详解

在 C 语言中，static 关键字不仅可以用来修饰变量，还可以用来修饰函数。在使用 static 关键字修饰变量时，我们称此变量为静态变量。

静态变量的存储方式与全局变量一样，都是静态存储方式。但这里需要特别说明的是，静态变量属于静态存储方式，属于静态存储方式的变量却不一定就是静态变量。例如，全局变量虽然属于静态存储方式，但并不是静态变量，它必须由 static 加以定义后才能成为静态全局变量。

考虑到可能会有不少读者对静态变量作用不太清楚，本节就来详细讨论一下它的主要作用。

隐藏与隔离的作用

上面已经阐述过，全局变量虽然属于静态存储方式，但并不是静态变量。全局变量的作用域是整个源程序，当一个源程序由多个源文件组成时，全局变量在各个源文件中都是有效的。

如果我们希望全局变量仅限于在本源文件中使用，在其他源文件中不能引用，也就是说限制其作用域只在定义该变量的源文件内有效，而在同一源程序的其他源文件中不能使用。这时，就可以通过在全局变量之前加上关键字 static 来实现，使全局变量被定义成为一个静态全局变量。这样就可以避免在其他源文件中引起的错误。也就起到了对其他源文件进行隐藏与隔离错误的作用，有利于模块化程序设计。

保持变量内容的持久性

有时候，我们希望函数中局部变量的值在函数调用结束之后不会消失，而仍然保留其原值。即它所占用的存储单元不释放，在下一次调用该函数时，其局部变量的值仍然存在，也就是上一次函数调用结束时的值。这时候，我们就应该将该局部变量用关键字 static 声明为“静态局部变量”。

当将局部变量声明为静态局部变量的时候，也就改变了局部变量的存储位置，即从原来的栈中存放改为静态存储区存放。这让它看起来很像全局变量，其实静态局部变量与全局变量的主要区别就在于可见性，静态局部变量只在其被声明的代码块中是可见的。

对某些必须在调用之间保持局部变量的值的子程序而言，静态局部变量是特别重要的。如果没有静态局部变量，则必须在这类函数中使用全局变量，由此也就打开了引入副作用的大门。使用静态局部变量最好的示例就是实现统计次数的功能，如下面示例所示。

```
01.  #include <stdio.h>
02.  void count();
03.  int main(void)
```

```
{
    int i=0;
    for (i = 0;i <= 5;i++)
    {
            count();
    }
    return 0;
}
void count()
{
    /*声明一个静态局部变量*/
    static num = 0;
    num++;
    printf("%d\n",num);
}
```

在该代码中，我们通过在 count() 函数里声明一个静态局部变量 num 来作为计数器。因为静态局部变量是在编译时赋初值的，且只赋初值一次，在程序运行时它已有初值。以后在每次调用函数时就不再重新赋初值，而是保留上次函数调用结束时的值。这样，count() 函数每次被调用的时候，静态局部变量 num 就会保持上一次调用的值，然后再执行自增运算，这样就实现了计数功能。同时，它又避免了使用全局变量。

通过上面的示例，我们可以得出静态局部变量一般的使用场景，如下所示：

- 需要保留函数上一次调用结束时的值。
- 如果初始化后，变量只会被引用而不会改变其值，则这时用静态局部变量比较方便，以免每次调用时重新赋值。

默认初始化为 0

在静态数据区，内存中所有的字节默认值都是 0x00。静态变量与全局变量也一样，它们都存储在静态数据区中，因此其变量的值默认也为 0。演示示例如下所示。

```
#include <stdio.h>
static int g_x;
int g_y;
int main(void)
{
    static int x;
    printf("g_x：%d\ng_y：%d\nx：%d",g_x,g_y,x);
    return 0;
}
```

运行结果为：

```
g_x：0
g_y：0
x：0
```