# C/C++中static关键字详解

静态变量作用范围在一个文件内，程序开始时分配空间，结束时释放空间，默认初始化为0，使用时可以改变其值。

静态变量或静态函数只有本文件内的代码才能访问它，它的名字在其它文件中不可见。

用法**1**：函数内部声明的**static**变量，可作为对象间的一种通信机制

如果一局部变量被声明为static，那么将只有唯一的一个静态分配的对象，它被用于在该函数的所有调用中表示这个变量。这个对象将只在执行线程第一次到达它的定义使初始化。

用法**2**：局部静态对象

对于局部静态对象，构造函数是在控制线程第一次通过该对象的定义时调用。在程序结束时，局部静态对象的析构函数将按照他们被构造的相反顺序逐一调用，没有规定确切时间。

用法**3**：静态成员和静态成员函数

如果一个变量是类的一部分，但却不是该类的各个对象的一部分，它就被成为是一个static静态成员。一个static成员只有唯一的一份副本，而不像常规的非static成员那样在每个对象里各有一份副本。同理，一个需要访问类成员，而不需要针对特定对象去调用的函数，也被称为一个static成员函数。

类的静态成员函数只能访问类的静态成员(变量或函数)。

进一步详细解释如下：

1.先来介绍它的第一条也是最重要的一条：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:

```
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。虽然这种用法不常见，但我还是举一个例子。

```
#include <stdio.h>

 int fun(void){

static int count = 10; // 事实上此赋值语句从来没有执行过

 return count--;

 }

 int count = 1;
```

```
int main(void) {
printf("global\t\tlocal static\n");
 for(; count <= 10; ++count)
    printf("%d\t\t%d\n", count, fun());
 return 0; }
```

程序的运行结果是：

global local static
1 10
2 9
3 8
4 7
5 6
6 5
7 4
8 3
9 2
10 1

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。比如初始化一个稀疏矩阵，我们可以一个一个地把所有元素都置0，然后把不是0的几个元素赋值。如果定义成静态的，就省去了一开始置0的操作。再比如要把一个字符数组当字符串来用，但又觉得每次在字符数组末尾加'\0'太麻烦。如果把字符串定义成静态的，就省去了这个麻烦，因为那里本来就是'\0'。不妨做个小实验验证一下。

```
#include <stdio.h>
 int a;
int main(void){
int i;
static char str[10];
printf("integer: %d; string: (begin)%s(end)", a, str);
 return 0;
 }
```

程序的运行结果如下integer： 0； string： （begin）（end）

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.

4. 用static声明的函数和变量小结

static 声明的变量在C语言中有两方面的特征：

1)、变量会被放在程序的全局存储区中，这样可以在下一次调用的时候还可以保持原来的赋值。这一点是它与堆栈变量和堆变量的区别。

2)、变量用static告知编译器，自己仅仅在变量的作用范围内可见。这一点是它与全局变量的区别。

Tips:
A.若全局变量仅在单个C文件中访问，则可以将这个变量修改为静态全局变量，以降低模块间的耦合度；
B.若全局变量仅由单个函数访问，则可以将这个变量改为该函数的静态局部变量，以降低模块间的耦合度；
C.设计和使用访问动态全局变量、静态全局变量、静态局部变量的函数时，需要考虑重入问题；
D.如果我们需要一个可重入的函数，那么，我们一定要避免函数中使用static变量(这样的函数被称为：带"内部存储器"功能的的函数)
E.函数中必须要使用static变量情况:比如当某函数的返回值为指针类型时，则必须是static的局部变量的地址作为返回值，若为auto类型，则返回为错指针。

函数前加static使得函数成为静态函数。但此处"static"的含义不是指存储方式，而是指对函数的作用域仅局限于本文件(所以又称内部函数)。使用内部函数的好处是：不同的人编写不同的函数时，不用担心自己定义的函数，是否会与其它文件中的函数同名。

扩展分析：

术语static有着不寻常的历史.起初，在C中引入关键字static是为了表示退出一个块后仍然存在的局部变量。随后，static在C中有了第二种含义：用来表示不能被其它文件访问的全局变量和函数。为了避免引入新的关键字，所以仍使用static关键字来表示这第二种含义。最后，C++重用了这个关键字，并赋予它与前面不同的第三种含义：表示属于一个类而不是属于此类的任何特定对象的变量和函数(与Java中此关键字的含义相同)。

全局变量、静态全局变量、静态局部变量和局部变量的区别

变量可以分为：全局变量、静态全局变量、静态局部变量和局部变量。

(1) 按存储区域分，全局变量、静态全局变量和静态局部变量都存放在内存的静态存储区域，局部变量存放在内存的栈区。

(2) 按作用域分， 全局变量在整个工程文件内都有效；静态全局变量只在定义它的文件内有效；静态局部变量只在定义它的函数内有效，只是程序仅分配一次内存，函数返回后，该变量不会消失；局部变量在定义它的函数内有效，但是函数返回后失效。

全局变量(外部变量)的说明之前再冠以static就构成了静态的全局变量。全局变量本身就是静态存储方式，静态全局变量当然也是静态存储方式。这两者在存储方式上并无不同。这两者的区别虽在于非静态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，非静态的全局变量在各个源文件中都是有效的。 而静态全局变量则限制了其作用域，即只在定义该变量的源文件内有效，在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用，因此可以避免在其它源文件中引起错误。

从以上分析可以看出， 把局部变量改变为静态变量后是改变了它的存储方式即改变了它的生存期。把全局变量改变为静态变量后是改变了它的作用域， 限制了它的使用范围。

(1) static 函数与普通函数作用域不同。仅在本文件。只在当前源文件中使用的函数应该说明为内部函数(static)，内部函数应该在当前源文件中说明和定义。对于可在当前源文件以外使用的函数，应该在一个头文件中说明，要使用这些函数的源文件要包含这个头文件

(2) static全局变量与普通的全局变量有什么区别：static全局变量只初始化一次，防止在其他文件单元中被引用;

(3) static局部变量和普通局部变量有什么区别：static局部变量只被初始化一次，下一次依据上一次结果值;
(4) static函数与普通函数有什么区别：static函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝.
(5) 全局变量和静态变量如果没有手工初始化，则由编译器初始化为0。局部变量的值不可知。

5. C++的static

C++的static有两种用法：面向过程程序设计的static和面向对象程序设计中的static。前者应用于普通变量和函数，不涉及类；后者主要说明static在类中的作用。

**(1)**、面向过程设计中的**static**

1)、静态全局变量

在全局变量前，加上关键字static，该变量就被定义成为一个静态全局变量。我们先举一个静态全局变量的例子，如下：

```
//Example 1
#include <iostream.h>
void fn();
static int n; //定义静态全局变量
void main()
{
 n=20;
 cout<<n<<endl;
 fn();
}
void fn()
{
 n++;
 cout<<n<<endl;
}
```

静态全局变量有以下特点：

i ） 该变量在全局数据区分配内存；
ii ） 未经初始化的静态全局变量会被程序自动初始化为0（自动变量的值是随机的，除非它被显式初始化）；
iii ）静态全局变量在声明它的整个文件都是可见的，而在文件之外是不可见的；

静态变量都在全局数据区分配内存，包括后面将要提到的静态局部变量。对于一个完整的程序，在内存中的分布情况如下图：

| 代码区 |
| --- |
| 全局数据区 |
| 堆区 |
| |

栈区

一般程序的由new产生的动态数据存放在堆区，函数内部的自动变量存放在栈区。自动变量一般会随着函数的退出而释放空间，静态数据（即使是函数内部的静态局部变量）也存放在全局数据区。全局数据区的数据并不会因为函数的退出而释放空间。细心的读者可能会发现，Example 1中的代码中将

```
static int n; //定义静态全局变量
```

改为

```
int n; //定义全局变量
```

程序照样正常运行。的确，定义全局变量就可以实现变量在文件中的共享，但定义静态全局变量还有以下好处：

1) 静态全局变量不能被其它文件所用；

2) 其它文件中可以定义相同名字的变量，不会发生冲突；

您可以将上述示例代码改为如下：

```
//Example 2
//File1
#include <iostream.h>
void fn();
static int n; //定义静态全局变量
void main()
{
 n=20;
 cout<<n<<endl;
 fn();
}
//File2
#include <iostream.h>
extern int n;
void fn()
{
 n++;
 cout<<n<<endl;
}
```

编译并运行Example 2，您就会发现上述代码可以分别通过编译，但运行时出现错误。试着将

```
static int n; //定义静态全局变量
```

改为

```
int n; //定义全局变量
```

再次编译运行程序，细心体会全局变量和静态全局变量的区别。

**(2)**、静态局部变量

在局部变量前，加上关键字static，该变量就被定义成为一个静态局部变量。 我们先举一个静态局部变量的例子，如下：

```
//Example 3
#include <iostream.h>
void fn();
void main()
{
 fn();
 fn();
 fn();
}
```

```
void fn()
{
 static n=10;
 cout<<n<<endl;
 n++;
}
```

通常，在函数体内定义了一个变量，每当程序运行到该语句时都会给该局部变量分配栈内存。但随着程序退出函数体，系统就会收回栈内存，局部变量也相应失效。但是有时候我们需要在两次调用之间对变量的值进行保存。通常的想法是定义一个全局变量来实现。但这样一来，变量已经不再属于函数本身了，不再仅受函数的控制，给程序的维护带来不便。　　静态局部变量正好可以解决这个问题。静态局部变量保存在全局数据区，而不是保存在栈中，每次的值保持到下

分类: C, C++



chao_yu
关注 - 0
粉丝 - 197
+加关注



« 上一篇：C/C++中作用域详解
» 下一篇：ubuntu清理系统垃圾与备份

posted on 2010-07-14 17:53 chao_yu 阅读(38738) 评论(3) 编辑 收藏

## 评论

**coding_monkey** | 2014-01-07 19:51　#1楼

学习了，但是博主main.c中有个小错误，就是也要extern void msg();

支持(0)　反对(1)

**zhanlaomeng** | 2014-07-27 18:49　#2楼

@coding_monkey
默认就有的，不必加的

支持(0)　反对(0)

**AnnPeter** | 2014-12-31 21:14　#3楼

博主 你举得第一个例子在.c文件是可以看见的，.cpp文件就不行了 这是为什么额？？

支持(0)　反对(0)

刷新评论　刷新页面　返回顶部

· Twitter拟推30秒视频和群聊私信功能
· 谷歌光纤网络覆盖范围新增四座城市
· 亚马逊会员每年网购花销是非会员的两倍多
» 更多新闻...



最新知识库文章：
· 浅析数据化设计思维
· 门户级UGC系统的技术进化路线
· 亿级用户下的新浪微博平台架构
· 技术团队的情绪与效率
· 关于请求被挂起页面加载缓慢问题的追查
» 更多知识库文章...