转 C语言堆栈入门 —— 堆和栈的区别

2013年10月15日 00:01:21　阅读数：2429

在计算机领域，堆栈是一个不容忽视的概念，我们编写的C语言程序基本上都要用到。但对于很多的初学着来说，堆栈是一个很模糊的概念。堆栈：一种数据结构、一个在程序运行时用于存放的地方，这可能是很多初学者的认识，因为我曾经就是这么想的和汇编语言中的堆栈一词混为一谈。我身边的一些编程的朋友以及在网上看帖遇到的朋友中有好多也说不清堆栈，所以我想有必要给大家分享一下我对堆栈的看法，有说的不对的地方请朋友们不吝赐教，这对于大家学习会有很大帮助。

数据结构的栈和堆

首先在数据结构上要知道堆栈，尽管我们这么称呼它，但实际上堆栈是两种数据结构：堆和栈。

堆和栈都是一种数据项按序排列的数据结构。

栈就像装数据的桶或箱子

我们先从大家比较熟悉的栈说起吧，它是一种具有 后进先出 性质的数据结构，也就是说后存放的先取，先存放的后取。这就如同我们要取出放在箱子里面底下的东西（放入的比较早的物体），我们首先要移开压在它上面的物体（放入的比较晚的物体）。

堆像一棵倒过来的树

而堆就不同了，堆是一种 经过排序的树形数据结构 ，每个结点都有一个值。通常我们所说的堆的数据结构，是指二叉堆。堆的特点是根结点的值最小（或最大），且根结点的两个子树也是一个堆。由于堆的这个特性，常用来实现优先队列， 堆的存取是随意 ，这就如同我们在图书馆的书架上取书，虽然书的摆放是有顺序的，但是我们想取任意一本时不必像栈一样，先取出前面所有的书，书架这种机制不同于箱子，我们可以直接取出我们想要的书。

内存分配中的栈和堆

然而我要说的重点并不在这，我要说的堆和栈并不是数据结构的堆和栈，之所以要说数据结构的堆和栈是为了和后面我要说的堆区和栈区区别开来，请大家一定要注意。

下面就说说C语言程序内存分配中的堆和栈，这里有必要把 内存分配 也提一下，大家不要嫌我啰嗦，一般情况下程序存放在Rom或Flash中，运行时需要拷到内存中执行，内存会分别存储不同的信息。

内存中的栈区处于相对较高的地址以地址的增长方向为上的话，栈地址是向下增长的。

栈中分配局部变量空间，堆区是向上增长的用于分配程序员申请的内存空间。另外还有静态区是分配静态变量，全局变量空间的；只读区是分配常量和程序代码空间的；以及其他一些分区。

来看一个网上很流行的经典例子：

```
main.cpp
int a = 0; 全局初始化区
char *p1; 全局未初始化区
main()
{
int b; 栈
char s[] = "abc"; 栈
char *p2; 栈
char *p3 = "123456"; 123456\0在常量区，p3在栈上。
static int c =0； 全局（静态）初始化区
p1 = (char *)malloc(10);  堆
p2 = (char *)malloc(20);  堆
}
```

0.申请方式和回收方式不同

不知道你是否有点明白了，堆和栈的第一个区别就是申请方式不同：栈（英文名称是stack）是系统自动分配空间的，例如我们定义一个 char a； 系统会自动在栈上为其开辟空间 。而堆（英文名称是heap）则是 程序员根据需要自己申请的空间 ，例如malloc（10）；开辟十个字节的空间。由于 栈上的空间是自动分配自动回收的 ，所以栈上的数据的生存周期只是在函数的运行过程中，运行后就释放掉，不可以再访问。而 堆上的数据只要程序员不释放空间，就一直可以访问到 ，不过缺点是一旦忘记释放会造成内存泄露。还有其他的一些区别我认为网上的朋友总结的不错这里转述一下：

1.申请后系统的响应

栈 ：只要栈的剩余空间大于所申请空间，系统将为程序提供内存，否则将报异常提示栈溢出。

堆 ：首先应该知道 操作系统有一个记录空闲内存地址的链表 ，当系统收到程序的申请时，会遍历该链表，寻找第一个空间大于所申请空间的堆。

结点，然后将该结点从空闲结点链表中删除，并将该结点的空间分配给程序，另外，对于大多数系统，会在这块内存空间中的 首地址处记录本次分配的大小 ，这样，代码中的 delete语句才能正确的释放本内存空间。另外，由于找到的堆结点的大小不一定正好等于申请的大小，系统会自动的 将多余的那部分重新放入空闲链表中 。也就是说 堆会在申请后还要做一些后续的工作这就会引出申请效率的问题。

2.申请效率的比较

根据第0点和第1点可知。

栈 ：由系统自动分配，速度较快。但程序员是无法控制的。

堆 ：是由new分配的内存，一般速度比较慢，而且容易产生内存碎片,不过用起来最方便。

3.申请大小的限制

栈：在Windows下,栈是向低地址扩展的数据结构,是一块　连续的内存的区域 。这句话的意思是　栈顶的地址和栈的最大容量是系统预先规定好的 ,在 WINDOWS下，栈的大小是2M（也有的说是1M，总之是一个编译时就确定的常数），如果申请的空间超过栈的剩余空间时，将提示overflow。因此，能从栈获得的空间较小。
堆　：堆是向高地址扩展的数据结构，是不连续的内存区域。这是由于系统是用链表来存储的空闲内存地址的，自然是不连续的，而链表的遍历方向是由低地址向高地址。堆的大小受限于计算机系统中有效的虚拟内存。由此可见，堆获得的空间比较灵活，也比较大。

4.堆和栈中的存储内容

由于栈的大小有限，所以用子函数还是有物理意义的，而不仅仅是逻辑意义。

栈 ： 在函数调用时,第一个进栈的是　主函数中函数调用后的下一条指令 （函数调用语句的下一条可执行语句）的地址，然后是函数的各个参数，在大多数的C编译器中，参数是由右往左入栈的，然后是 函数中的局部变量 。注意静态变量是不入栈的。
当本次函数调用结束后，局部变量先出栈，然后是参数，最后栈顶指针指向最开始存的地址，也就是主函数中的下一条指令，程序由该点继续运行。
堆 ：一般是在堆的头部用一个字节存放堆的大小。堆中的具体内容有程序员安排。

关于存储内容还可以参考 **这道题** 。这道题还涉及到局部变量的存活期。

5.存取效率的比较

```
char s1[] = "aaaaaaaaaaaaaa";
char *s2 = "bbbbbbbbbbbbbbbb";
```

aaaaaaaaaaa是在运行时刻赋值的；放在栈中。
而bbbbbbbbbb是在编译时就确定的；放在堆中。
但是，在以后的存取中，在栈上的数组比指针所指向的字符串(例如堆)快。
比如：

```
#include
void main()
{
char a = 1;
char c[] = "1234567890";
char *p ="1234567890";
a = c[1];
a = p[1];
return;
}
```

对应的汇编代码

```
10: a = c[1];
00401067 8A 4D F1 mov cl,byte ptr [ebp-0Fh]
0040106A 88 4D FC mov byte ptr [ebp-4],cl
11: a = p[1];
0040106D 8B 55 EC mov edx,dword ptr [ebp-14h]
00401070 8A 42 01 mov al,byte ptr [edx+1]
00401073 88 45 FC mov byte ptr [ebp-4],al
```

关于堆和栈区别的比喻

堆和栈的区别可以引用一位前辈的比喻来看出：
使用栈就象我们去饭馆里吃饭，只管点菜（发出申请）、付钱、和吃（使用），吃饱了就走，不必理会切菜、洗菜等准备工作和洗碗、刷锅等扫尾工作，他的好处是快捷，但是自由度小。
使用堆就象是自己动手做喜欢吃的菜肴，比较麻烦，但是比较符合自己的口味，而且自由度大。比喻很形象，说的很通俗易懂，不知道你是否有点收获。

文章标签： 栈　堆　区别　c
个人分类： 纯编程