**C++ 堆栈与函数调用**

时间:2009-03-03 16:52来源:未知 作者:看看Linux 点击:859次

一 C++程序内存分配

1) 在栈上创建。在执行函数时，函数内局部变量的存储单元都在栈上创建，函数执行结束时这些存储单元自动被释放。栈内存分配运算内置于处理器的指令集中，一般使用寄存器来存取，效率很高，但是分配的内存容量有限。  
2) 从堆上分配，亦称动态内存分配。程序在运行的时候用malloc或new申请任意多少的内存，程序员自己负责在何时用free或delete来释放内存。动态内存的生存期由程序员自己决定，使用非常灵活。  
3) 从静态存储区域分配。内存在程序编译的时候就已经分配好，这块内存在程序的整个运行期间都存在。例如全局变量，static变量。  
4) 文字常量分配在文字常量区，程序结束后由系统释放。  
5）程序代码区。  
经典实例：

#include<string>

inta=0;   //全局初始化区  
char \*p1;  //全局未初始化区  
 voidmain()  
{  
   intb;//栈  
   chars[]="abc";  //栈  
   char \*p2;        //栈  
   char \*p3="123456";  //123456\0在常量区，p3在栈上。  
   static intc=0;  //全局（静态）初始化区  
    p1=(char\*)malloc(10);  
    p2=(char\*)malloc(20);  //分配得来得10和20字节的区域就在堆区。  
    strcpy(p1,"123456");  //123456\0放在常量区，编译器可能会将它与p3所向"123456\0"优化成一个地方。  
}

二 三种内存对象的比较  
　　栈对象的优势是在适当的时候自动生成，又在适当的时候自动销毁，不需要程序员操心；而且栈对象的创建速度一般较堆对象快，因为分配堆对象时，会调用operator new操作，operator new会采用某种内存空间搜索算法，而该搜索过程可能是很费时间的，产生栈对象则没有这么麻烦，它仅仅需要移动栈顶指针就可以了。但是要注意的是，通常栈空间容量比较小，一般是1MB～2MB，所以体积比较大的对象不适合在栈中分配。特别要注意递归函数中最好不要使用栈对象，因为随着递归调用深度的增加，所需的栈空间也会线性增加，当所需栈空间不够时，便会导致栈溢出，这样就会产生运行时错误。  
　　堆对象创建和销毁都要由程序员负责，所以，如果处理不好，就会发生内存问题。如果分配了堆对象，却忘记了释放，就会产生内存泄漏；而如 果已释放了对象，却没有将相应的指针置为NULL，该指针就是所谓的“悬挂指针”，再度使用此指针时，就会出现非法访问，严重时就导致程序崩溃。但是高效的使用堆对象也可以大大的提高代码质量。比如，我们需要创建一个大对象，且需要被多个函数所访问，那么这个时候创建一个堆对象无疑是良好的选择，因为我们通过在各个函数之间传递这个堆对象的指针，便可以实现对该对象的共享，相比整个对象的传递，大大的降低了对象的拷贝时间。另外，相比于栈空间，堆的容量要大得多。实际上，当物理内存不够时，如果这时还需要生成新的堆对象，通常不会产生运行时错误，而是系统会使用虚拟内存来扩展实际的物理内存。  
　　静态存储区。所有的静态对象、全局对象都于静态存储区分配。关于全局对象，是在main()函数执行前就分配好了的。其实，在main()函数中的显示代 码执行之前，会调用一个由编译器生成的\_main()函数，而\_main()函数会进行所有全局对象的的构造及初始化工作。而在main()函数结束之 前，会调用由编译器生成的exit函数，来释放所有的全局对象。比如下面的代码：

void main（void）  
{  
… …// 显式代码  
}

实际上，被转化成这样：

void main（void）  
{  
\_main（）; //隐式代码，由编译器产生，用以构造所有全局对象  
… … // 显式代码  
… …  
exit（） ; // 隐式代码，由编译器产生，用以释放所有全局对象  
}