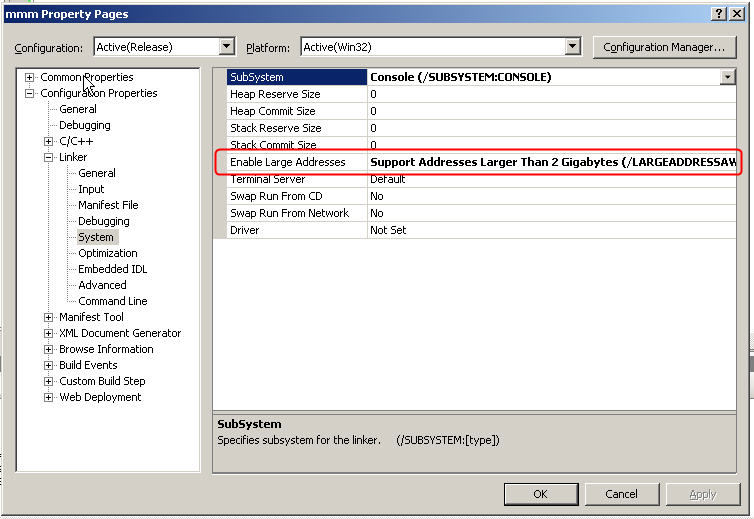
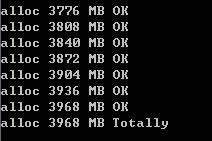
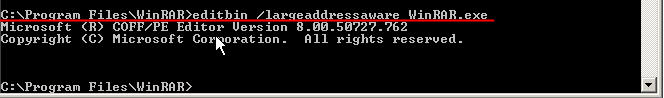
一般情况下，无论在32位系统还是64位系统下，一个Win32用户进程可用的内存空间只有2GB少一点。那么，假设一个Win32程序内存不够用了，又不想(或者没办法)编译成X64的，怎么办呢，MS提供了一种方法，Linker中的LargeAddressAware参数。  
启用它，可以获得额外的用户态内存。我在自己的X64系统下测试，如果没有启用，可以申请2GB不到；但是启动了这个参数，则可以申请将近4GB（因为还要受32位程序32位指针的限制)：  
  
  
程序很简单，如下：

**[cpp]** [view plain](http://blog.csdn.net/lff0305/article/details/7613741) [copy](http://blog.csdn.net/lff0305/article/details/7613741)

1. **int** \_tmain(**int** argc, \_TCHAR\* argv[])
2. {
3. **int** c = 0;
4. **while** (**true**) {
5. **void** \* p = malloc(1024 \* 1024 \* 32);
6. **if** (p != 0) {
7. c++;
8. printf("alloc %d MB OK\n", c \* 32);
9. } **else** {
10. **break**;
11. }
12. }
13. printf("alloc %d MB Totally\n", c \* 32);
14. getchar();
15. **return** 0;
16. }

运行结果是  
  
该程序如果不设置LargeAddressAware参数的话是1986MB。  
我自己没有测试32位的系统，但是查到的资料说需要设置/3GB的启动参数。  
对于已经存在的程序(没办法编译)，可以使用VC中带的editbin.exe工具把该标志位打开：  
  
这样这个程序也可以使用超过2GB的内存了。  
同样对于一个exe，可以使用dumpbin.exe来验证。  
如果dumpbin.exe /headers 文件名.exe  
显示 Application can handle large (>2GB) addresses  
那么说明该应用程序可以使用超过2GB的内存  
当然这种方法的缺点:  
1. 在64位系统上没办法使用超过4GB的内存；不过以前只能用2G，聊胜于无了；

2. 在32位系统上必须打开/3GB启动参数，为用户态程序预留3GB的内存

3. 对于带有自校验的程序，不适用，因为dumpbin相当于修改了这个exe，那么文件校验肯定是失败了；

总之最终的解决方法还是使用(或者编译出)X64的程序。