# 实验八 Windows内存管理

### 实验内容1 Windows内存结构

#### 背景知识简介

* Windows XP(32位版本)是32位的操作系统，它使计算机CPU可以用32位地址对32位内存块进行操作。内存中的每一个字节都可以用一个32位的指针来寻址。这样，最大的存储空间就是232字节或4000兆字节 (4GB) 。这样，在Windows下运行的每一个应用程序都认为能独占可能的4GB大小的空间。
* 而另一方面，实际上没有几台机器的RAM能达到4GB，更不必说让每个进程都独享4GB内存了。Windows在幕后将虚拟内存 (virtual memory，VM) 地址(逻辑地址空间中的地址)映射到了各进程的物理内存地址上。而所谓物理内存是指计算机的RAM和由Windows分配到用户驱动器根目录上的换页文件。物理内存完全由系统管理。

#### 实验目的和要求

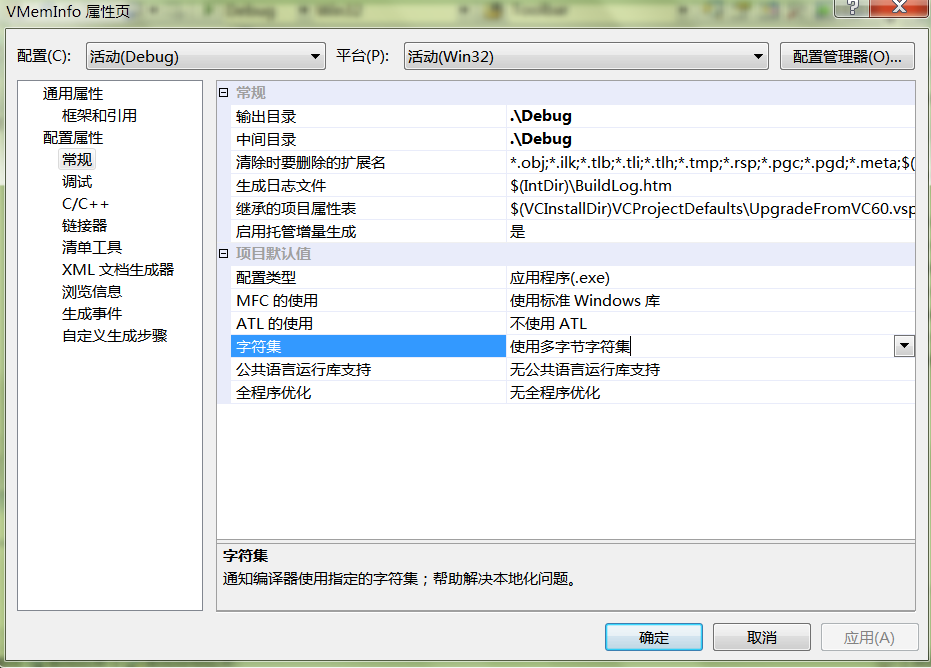
* 通过实验了解Windows 内存的使用，学习如何在应用程序中管理内存，体会Windows应用程序内存的简单性和自我防护能力
* 了解Windows的内存结构和虚拟内存的管理，进而了解进程堆和Windows为使用内存而提供的一些扩展功能

#### 实验环境

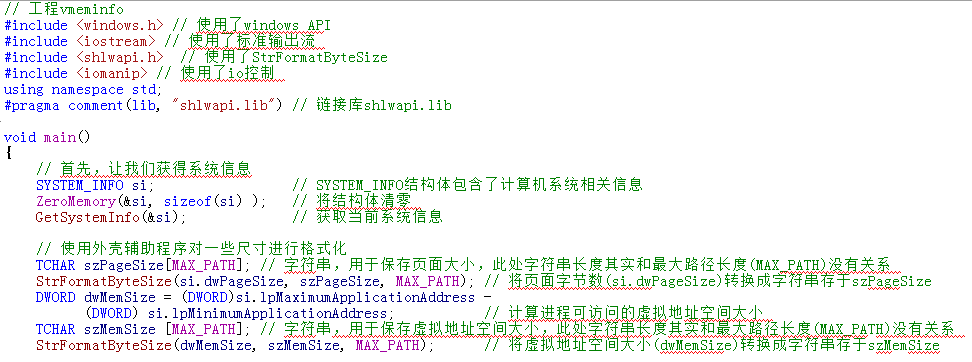
* 需要做以下准备：
  + 一台运行Windows操作系统的计算机
  + 计算机中需安装Visual C++

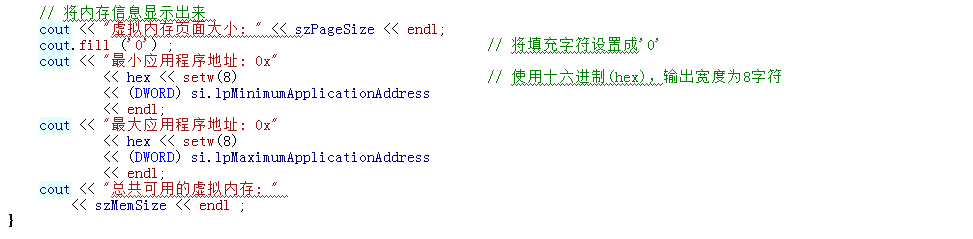
#### 实验步骤

新建一个空的Windows Console Application，项目名为XXXXMemInfo，其中XXXX为你的姓名拼音首字母和学号末两位。并新建一个C++源文件，命名任意。修改项目属性，将字符集设为多字节字符集(即不使用Unicode.因为程序中用到StrFormatByteSize这个API,需要用多字节字符集)：



Windows提供了一个API即GetSystemInfo() ，以便用户能获取系统的相关信息，其中包括虚拟内存的一些信息。以下源程序显示了如何调用该函数以及显示系统中当前内存的参数(具体内容见源程序)。在你的cpp文件中键入以下程序代码，并理解其代码含义。(参见vmeminfo.cpp)。





在4G的虚拟地址空间中，低端的2G是用户地址空间，高端的2G是系统地址空间。但为何显示可用虚拟内存不到2G呢？

提示:

可供应用程序使用的内存空间实际上已经减去了开头与结尾两个64KB的保护区。虚拟内存空间中的64KB保护区是防止编程错误的一种Windows方式。任何对内存中这一区域的访问 (读、写、执行) 都将引发一个错误陷井，从而导致错误并终止程序的执行。也就是说，假如用户有一个NULL指针 (地址为0) ，但仍试图在此之前很近的地址处使用另一个指针，这将因为试图从更低的保留区域读写数据，从而产生意外错误并终止程序的执行。

32位应用程序进程空间地址

0x00000000 到 0x0fffffff 包含内核。

0x10000000 到 0x1fffffff 包含应用程序文本。

0x20000000 到 0x2fffffff 包含应用程序数据、进程堆和应用程序堆栈。

0x30000000 到 0xcfffffff 可供共享内存或 mmap 服务使用。

0xd0000000 到 0xdfffffff 包含共享库文本。

0xe0000000 到 0xefffffff 可供共享内存或 mmap 服务使用。

0xf0000000 到 0xffffffff 包含应用程序共享库数据。

SYSTEM\_INFO结构体其实还包括了很多其他系统信息，试着增加代码，显示其他相关信息：

用StrFormatByteSize将si.dwAllocationGranularity(内存分配粒度，即在分配内存时至少要分配多少内存)转换并输出。

输出系统CPU核心数(si.dwNumberOfProcessors)。

截图运行结果。截图需包含你的工程名称。

### 实验内容2 Windows虚拟内存

#### 背景知识简介

* 在Windows XP环境下，4GB的虚拟地址空间被划分成两个部分：低端2GB提供给用户进程使用，高端2GB提供给系统使用。这意味着用户的应用程序代码，包括DLL以及进程使用的各种数据等，都装在用户进程地址空间内 (低端2GB)
* 内存页面可以有三种状态：未用（Free）、保留(Reserved)和提交（Committed）。一个未用的页面是指该页面未被保留或是提交，对一个进程来讲一个未用的页面是不可访问的，访问这样的页面将导致访问违例。进程可以要求系统保留一些页面以备后用，系统返回一段保留的地址给进程，但是这些地址同样是不可访问的，进程若想使用这段地址空间，使用前必须先提交。只有一个提交的页面才是一个真正可以访问的页面。另外，这三个状态的两两之间都是可以相互转化的。相关的API函数有VirtualAlloc、VirtualAllocEx、VirtualFree、VirtualFreeEx等
* 与虚拟内存区相关的访问权限告知系统进程可在内存中进行何种类型的操作。例如，用户不能在只有PAGE\_READONLY权限的区域上进行写操作或执行程序；也不能在只有PAGE\_EXECUTE权限的区域里进行读、写操作。而具有PAGE\_ NOACCESS权限的特殊区域，则意味着不允许进程对其地址进行任何操作
* 在进程装入之前，整个虚拟内存的地址空间都被设置为只有PAGE\_NOACCESS权限的自由区域。当系统装入进程代码和数据后，才将内存地址的空间标记为已调配区，并将诸如EXECUTE、READWRITE和READONLY的权限与这些区域相关联
* Windows还提供了一整套能使用户精确控制应用程序的虚拟地址空间的虚拟内存API
* 提供虚拟内存分配功能的是VirtualAlloc() API。该API支持用户向系统要求新的虚拟内存或改变已分配内存的当前状态。用户若想通过VirtualAlloc() 函数使用虚拟内存，可以采用两种方式通知系统：
  + 简单地将内存内容保存在地址空间内
  + 请求系统返回带有物理存储区 (RAM的空间或换页文件) 的部分地址空间
* 用户可以用flAllocation Type参数 (commit和reserve) 来定义这些方式，用户可以通知Windows按只读、读写、不可读写、执行或特殊方式来处理新的虚拟内存
* 与VirtualAlloc() 函数对应的是VirtualFree() 函数，其作用是释放虚拟内存中的已调配页或保留页。用户可利用dwFree Type参数将已调配页修改成保留页属性
* VirtualProtect() 是VirtualAlloc() 的一个辅助函数，利用它可以改变虚拟内存区的保护规范

#### 实验目的和要求

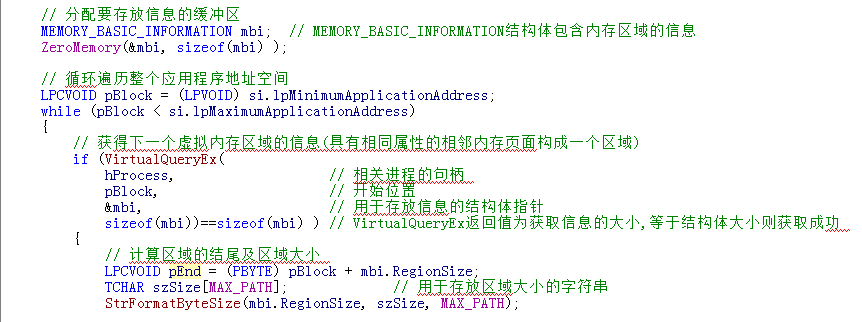
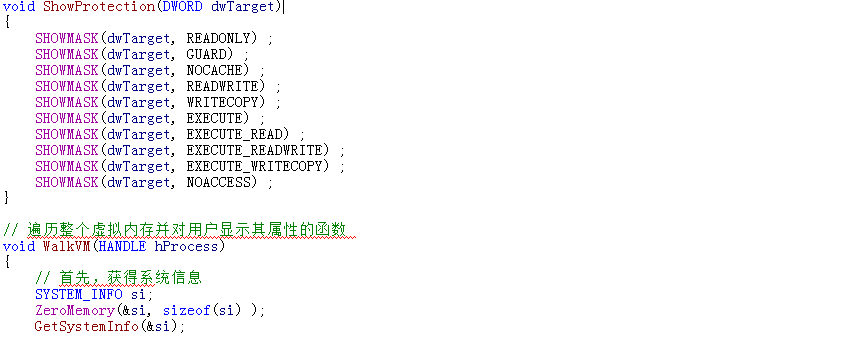
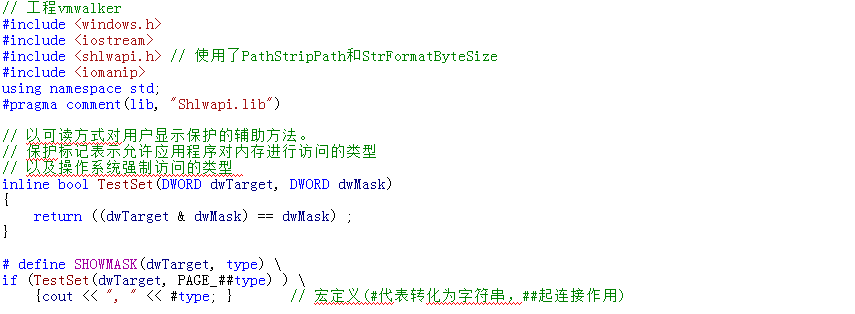
* 通过实验了解Windows内存的使用，学习如何在应用程序中管理内存，体会Windows应用程序内存的简单性和自我防护能力
* 学习检查虚拟内存空间或对其进行操作
* 了解Windows的内存结构和虚拟内存的管理，进而了解进程堆和Windows为使用内存而提供的一些扩展功能

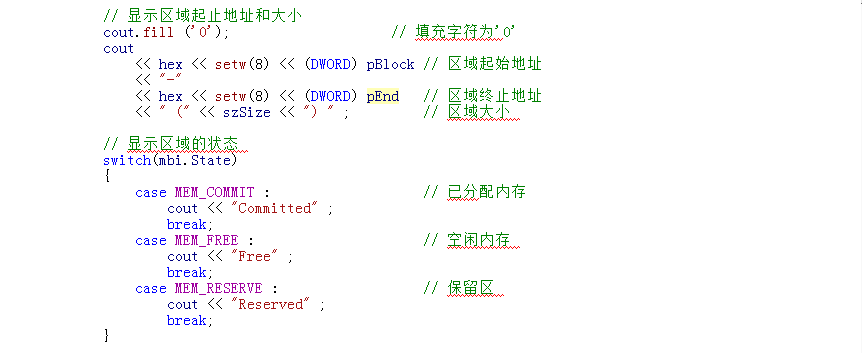
#### 实验环境

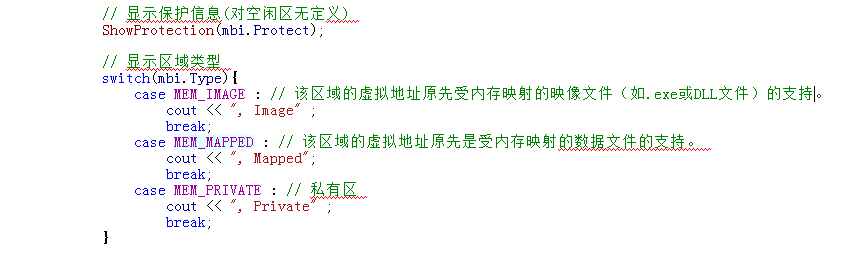
* 需要做以下准备
  + 一台运行Windows操作系统的计算机
  + 计算机中需安装Visual C++

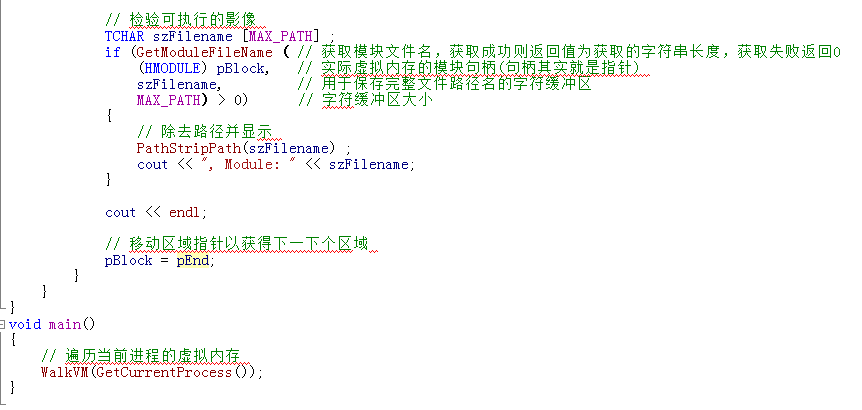
#### 实验内容与步骤

* 1. 新建一个空的Windows Console Application，项目名为XXXXvmwalker，其中XXXX为你的姓名拼音首字母和学号末两位。并新建一个C++源文件，命名任意。修改项目属性，不使用Unicode字符集。输入以下源程序代码，该程序用于虚拟内存的检测，使用了VirtualQueryEx()函数来检查虚拟内存空间(具体内容见源程序1)









2. 理解上述代码，思考问题：

1. 进程的地址空间中的内存状态可分为哪几种？

2. 其中已分配内存的区域又可分为哪几类？

答案写入实验报告。