Windows 操作系统

C/C++ 程序实验

姓名：_____陈展博_____

学号：_____1221001003__

班级：_____计科 1 班____

院系：_____信工_____

_____2024_____年___11__月__13___日

实验六　Windows 虚拟内存

一、背景知识

二、实验目的

三、工具/准备工作

四、实验内容与步骤

1. 虚拟内存的检测

清单 6-1 所示程序使用 VirtualQueryEX() 函数来检查虚拟内存空间。

步骤 1：登录进入 Windows 。

步骤 2：在“开始”菜单中单击“程序-Microsoft Visual Studio Code”。

步骤 3：新建项目名为“6-1”，并且新建项“6-1.cpp”。

清单 6-1 中显示一个 WalkVM() 函数开始于某个进程可访问的最低端虚拟地址处，并在其中显示各块虚拟内存的特性。虚拟内存中的块由 VirtualQueryEx() API 定义成连续快或具有相同状态 (自由区、已调配区等等) 的内存，并分配以一组统一的保护标志 (只读、可执行等等) 。

步骤 4：在代码宏定义里添加 #define _CRT_SECURE_NO_WARNINGS

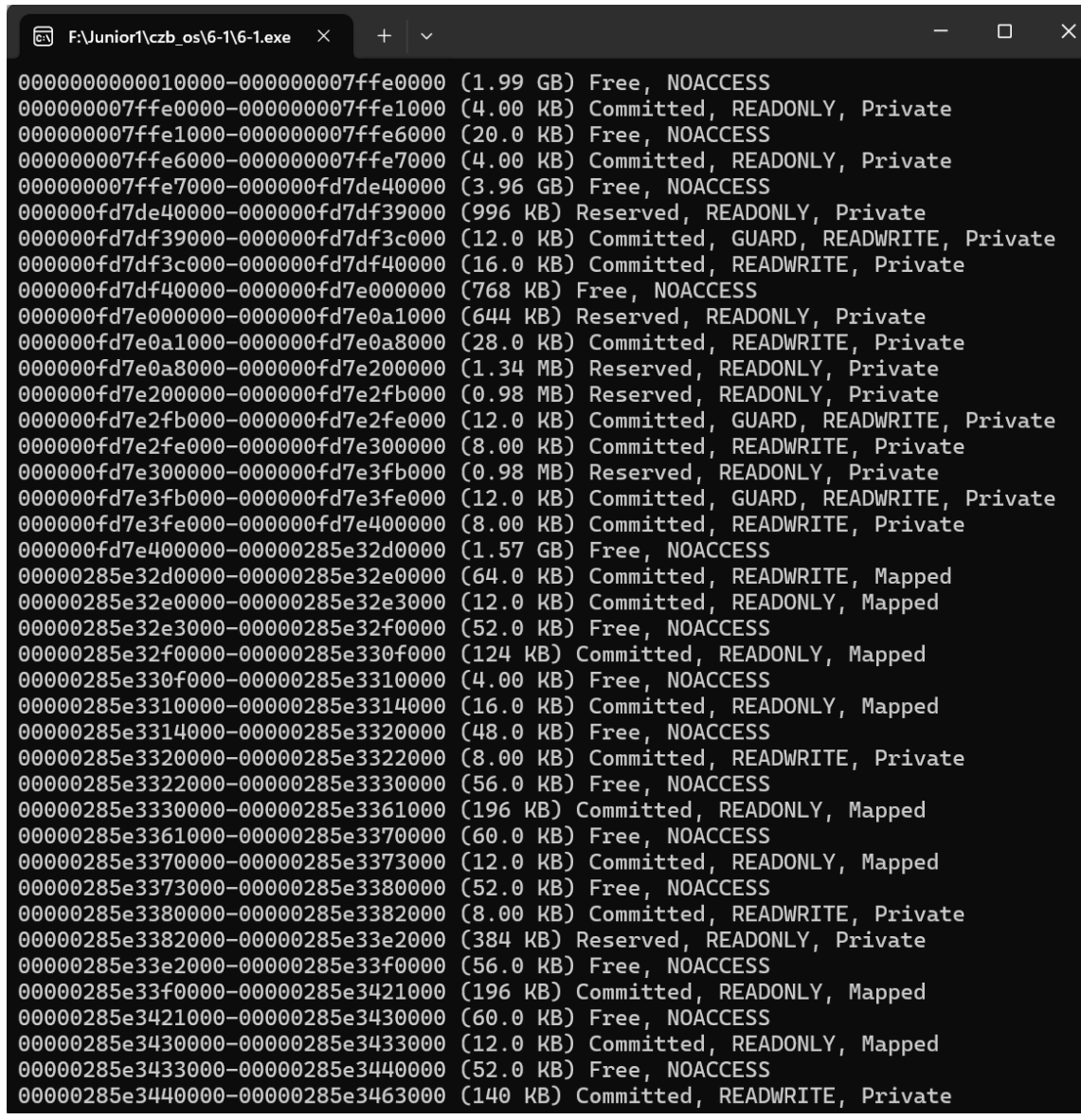
步骤 5：按“F5”开始调试，注意路径里不要含有中文。

步骤 6：按暂停按钮可暂停程序的执行，按终止按钮可终止程序的执行。

操作能否正常进行？如果不行，则可能的原因是什么？

操作能正常进行，如果不行可能是因为路径中包含中文或代码中包含中文字符。

1) 分析运行结果 (如果运行不成功，则可能的原因是什么？) 。



图片 1 运行结果如图

按 committed、reserved、free 等三种虚拟地址空间分别记录实验数据。其中“描述”是指对该组数据的简单描述，例如，对下列一组数据：

00010000 – 00012000/xp <8.00KB> Committed, READWRITE, Private

可描述为：具有 READWRITE 权限的已调配私有内存区。

将系统当前的自由区 (free) 虚拟地址空间填入表 6-3 中。

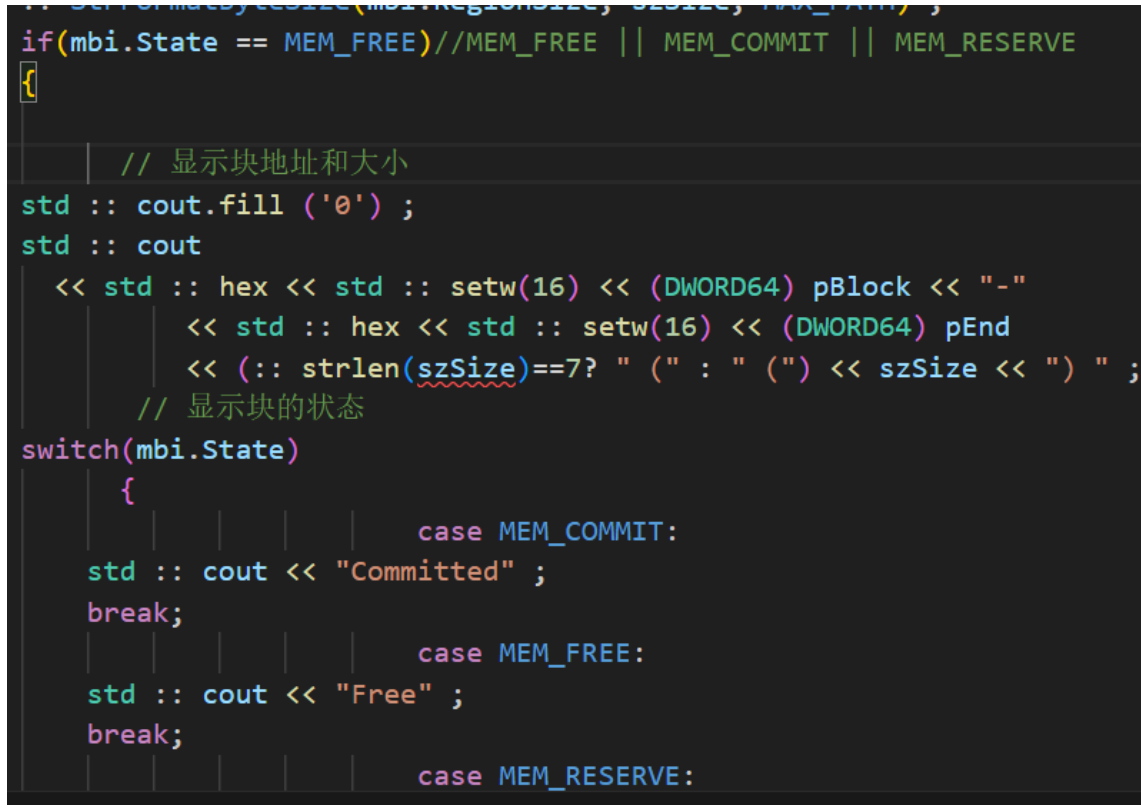
表 6-3 实验记录

地 址	大小	虚拟地址空间类型	访问权限	描 述

0000000000010000-000000007ffe0000	(1.99 GB)	free	NOACCESS	未分配内存空间的自由区，无法访问
000000007ffe1000-000000007ffe6000	(20.0 KB)	free	NOACCESS	未分配内存空间的自由区，无法访问
000000007ffe7000-000000fd7de40000	(3.96 GB)	free	NOACCESS	未分配内存空间的自由区，无法访问
000000fd7df40000-000000fd7e000000	(768 KB)	free	NOACCESS	未分配内存空间的自由区，无法访问
000000fd7e400000-00000285e32d0000	(1.57 GB)	free	NOACCESS	未分配内存空间的自由区，无法访问
00000285e330f000-00000285e3310000	(4.00 KB)	free	NOACCESS	未分配内存空间的自由区，无法访问
:	:	:	:	:
00007ffa2bb27000-00007fffffff0000	(3.31 GB)	free	NOACCESS	未分配内存空间的自由区，无法访问

提示：详细记录实验数据在实验活动中是必要的，但想想是否可以简化记录的办法？

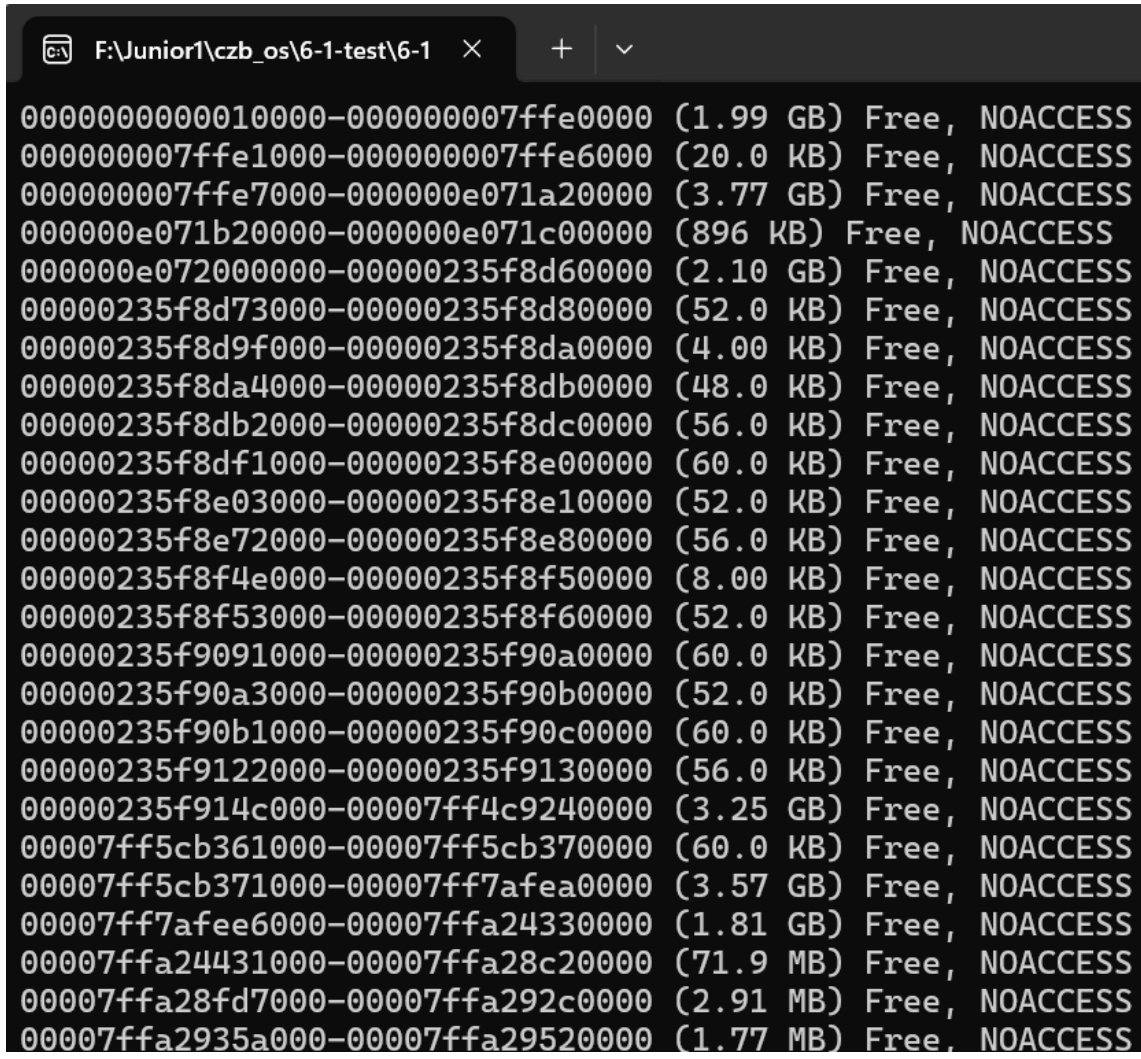
可以将代码中加入约束条件使得程序每次只打印同一种虚拟空间地址类型的记录，方便记录。



```
if(mbi.State == MEM_FREE)//MEM_FREE || MEM_COMMIT || MEM_RESERVE
{

    // 显示块地址和大小
std :: cout.fill ('0') ;
std :: cout
  << std :: hex << std :: setw(16) << (DWORD64) pBlock << "-"
        << std :: hex << std :: setw(16) << (DWORD64) pEnd
        << (:: strlen(szSize)==7? " (" : " (") << szSize << ") " ;
      // 显示块的状态
switch(mbi.State)
     {
                        case MEM_COMMIT:
    std :: cout << "Committed" ;
    break;
                        case MEM_FREE:
    std :: cout << "Free" ;
    break;
                        case MEM_RESERVE:
```

图片 2 附加条件（三选一使得每次只输出一种类型）



图片 3 同一类型记录显示

将系统当前的已调配区 (committed) 虚拟地址空间填入表 6-4 中。

表 6-4 实验记录

地 址	大小	虚拟地址空间类型	访问权限	描 述
000000007ffe0000-000000007ffe1000	(4.00 KB)	committed	READONLY	具有 READONLY（只读）权限的已调配私有内存区。
000000007ffe6000-000000007ffe7000	(4.00 KB)	committed	READONLY	具有 READONLY（只读）权限的已调配私有内存区。
000000fd7df39000-	(12.0 KB)	committed	GUARD,	具有保护（Guard）特性和

000000fd7df3c000			READWRITE	READWRITE（读写）权限的已调配私有内存区。
000000fd7df3c000-000000fd7df40000	(16.0 KB)	committed	READWRITE	具有 READWRITE（读写）权限的已调配私有内存区。
32808000-3280f000	(28.0 KB)	committed	READWRITE	具有 READWRITE（读写）权限的已调配私有内存区。
000000fd7e2fb000-000000fd7e2fe000	(12.0 KB)	committed	GUARD, READWRITE	具有保护（Guard）特性和 READWRITE 权限的已调配私有内存区。
:	:	:	:	:
00007ffa2ba9c000-00007ffa2bb27000	(556 KB)	committed	READONLY	包含映像文件的具有 READONLY（只读）权限的内存区域。

将系统当前的保留区 (reserved) 虚拟地址空间填入表 6-5 中。

表 6-5　实验记录

地　址	大小	虚拟地址空间类型	访问权限	描　述
000000fd7de40000-000000fd7df39000	(996 KB)	reserved	READONLY	预留的私有内存区域，设为只读权限。
000000fd7e000000-000000fd7e0a1000	(664 KB)	reserved	READONLY	预留的私有内存区域，设为只读权限。
000000fd7e0a8000-000000fd7e200000	(1.34 MB)	reserved	READONLY	预留的私有内存区域，设为只读权限。
000000fd7e200000-000000fd7e2fb000	(0.98 MB)	reserved	READONLY	预留的私有内存区域，设为只读权限。

000000fd7e300000- 000000fd7e3fb000	(0.98 MB)	reserved	READONLY	预留的私有内存区域，设为只读权限。
00000285e3463000- 00000285e3540000	(884 KB)	reserved	READONLY	预留的私有内存区域，设为只读权限。
:	:	:	:	:
00007ff4b7b90000- 00007ff5b7bb0000	(128 KB)	reserved	READONLY	预留的私有内存区域，设为只读权限。

2) 从上述输出结果，对照分析 6-1 程序，请描述程序运行的流程：

程序先通过::GetSystemInfo 获取系统信息，其中包括进程可以应用的最大和最小地址范围，再初始化分配要存放信息的缓存区。将常量长指针 pBlock 初始化为进程的最小地址，并进入 while 循环，当 pBlock 没有超出进程的最大应用地址时，遍历虚拟内存。每次进入 while 循环使用 VirtualQueryEx 获取内存块信息，并利用缓冲区 mbi 大小计算出末尾指针的地址，通过 mbi.State 获取内存状态（Commited/Free/Reserved）；再显示保护属性如果 mbi.Protect 为 0 并且状态不是 Free，则默认设置为 PAGE_READONLY；通过 mbi.Type 显示内存类型（Image/Mapped/Private）；如果内存区域属于某个模块，输出可执行的影像的相关信息。

2. 虚拟内存操作

清单 6-2 的示例显示了如何分配一个大容量空间，将物理存储委托给其中的很小一部分（千分之一）并加以使用。

步骤 7：新建项目名为“6-2”，并且新建项“6-2.cpp”。

步骤 8：按“F5”开始调试，注意路径里不要含有中文。

步骤 9：按暂停按钮可暂停程序的执行，按终止按钮可终止程序的执行。

操作能正常进行，如果不行可能是因为路径中包含中文或代码中包含中文字符。

运行结果（如果运行不成功，则可能的原因是什么？）：

1) 文件中含有中文双引号，导致运行不成功。

2) 文件路径中含有中文，导致运行不成功。

3)

图片 4 运行结果

对照运行结果，分析程序 6-2。为了给数据库保留 1GB 的段地址空间，清单 6-2 给出了内存分配的四种方法。

- **第一种技术**

即程序中说明为使用内存分配来获得 1GB 块的程序段，该段程序试图利用标准 C 中的 malloc() 函数，从已经已调配的小内存区获得内存。从运行结果看，这种技术成功了吗？

这种技术成功了，分配的 1GB 内存块被成功以指定长度的零数字填充。

- **第二种技术**

即程序中说明为使用虚拟分配以获得物理 1GB 块的程序段，该段程序试图通过 VirtualAlloc() ，然后利用物理备用内存将整个块分配到虚拟内存空间的任何位置。这种技术只对拥有 1GB 以上的 RAM 且都有换页文件的计算机可行。从运行结果看，这种技术成功了吗？这种技术成功了，使用虚拟分配获得的 1GB 物理内存块被成功以指定长度的零数字填充。

- **第三种技术**

即程序中说明为使用虚拟分配以获得虚拟 1GB 块的程序段，该段程序利用 VirtualAlloc() ，如果函数成功，则获得大块内存，但不将任何物理内存调配到此块中。从运行结果看，这种技术成功了吗？这种技术成功了，因为只分配了虚拟的 1GB 内存块而未给申请的内存空间调配物理存储，所以无法以指定长度的零数字填充内存块。

- **第四种技术**

即程序中说明为使用虚拟分配调配获得虚拟 1GB 块，再为其调配 1MB 物理存储的程序段，该段程序保留 1GB 的内存区，然后将物理内存调配给其中的很小一部分 (1MB) 。这就是清单 6-2 介绍的处理一个假想的数据库应用程序的方法：保留整个块，然后按要求在其一小部分内进行读操作，让系统将用过的区域换页到磁盘中。

利用 VirtualLock() API，Windows 可用来在自己的进程空间中控制虚拟内存的行为。这个

函数与其成对的 VirtualUnlock() 阻止或允许一块内存从物理 RAM 中换页和换页到页面文件中。这样就会通知系统有一段特定的内存区要求对用户作出强烈的响应，所以系统不应将其移出 RAM。当然，如果要将整个虚拟内存空间锁定，系统就会停留于试图将系统中工作内存的每一小块换页到磁盘。