操作系统
内存管理实验报告

编译环境：Microsoft Visual Studio 2019

Windows SDK **版本：**10.0

平台工具集：Visual Studio 2019（v142）

语言：C/C++

一、 实验目的：

在本次实验中，需要从不同的侧面了解Windows　的虚拟内存机制。在Windows操作系统中，可以通过一些API 操纵虚拟内存。主要需要了解以下几方面：

1、Windows　虚拟存储系统的组织

2、如何控制虚拟内存空间

3、如何编写内存追踪和显示工具

4、详细了解与内存相关的API 函数的使用

二、 实验需求：

编写一个包含两个线程的进程，一个线程用于模拟内存分配活动，一个线程用于跟踪第一个线程的内存行为。模拟内存活动的线程可以从一个文件中读出要进行的内存操作，每个内存操作包含如下内容：

时间：开始执行的时间；

块数：分配内存的粒度；

操作：包括保留一个区域、提交一个区域、释放一个区域、回收一个区域以及锁与解锁一个区域；可以将这些操作编号，存放于文件中。

大小：指块的大小；

访问权限：共五种 PAGE_READONLY、PAGE_READWRITE、PAGE_EXCUTE、PAGE_EXECUTE_READ 和 PAGE_ EXECUTE_READWRITE。可以将这些权限编号，存放于文件中。跟踪线程将页面大小、已使用的地址范围、物理内存总量以及虚拟内存总量等信息显示出来。

三、 实验原理：

内存管理是 Windows 执行体的一部分，位于 Ntoskrnl.exe 文件中，是整个操作系统的重要组成部分。

物理内存是固定的，内存条的容量多大，物理内存就有多大。但是如果程序运行很多或者程序本身很大的话，就会导致大量的物理内存占用，甚至导致物理内存消耗殆尽。

虚拟内存就是在硬盘上划分一块页面文件，充当内存。当程序在运行时，有一部分资源还没有用上或者同时打开几个程序却只操作其中一个程序时，系统没必要将程序所有的资源都放在物理内存中。于是，系统将这些暂时不用的资源放在虚拟内存上，等到需要时在调出来用。这也是虚拟内存的优点：需要的时候才真正分配内存。

默认情况下，32 位 Windows 上每个用户进程可以占有 2GB 的私有地址空间，操作系统

占有剩下的 2GB。Windows 在 x86 体系结构上利用二级页表结构来实现虚拟地址向物理地址的变换。一个 32 位虚拟地址被解释为三个独立的分量——页目录索引、页表索引和字节索引——它们用于找出描述页面映射结构的索引。页面大小及页表项的宽度决定了页目录和页表索引的宽度。比如，在 x86 系统中，因为一页包含 4096 字节，于是字节索引被确定为 12 位宽（2^12=4096）。

应用程序有三种使用内存方法：

以页为单位的虚拟内存分配方法，适合于大型对象或结构数组；

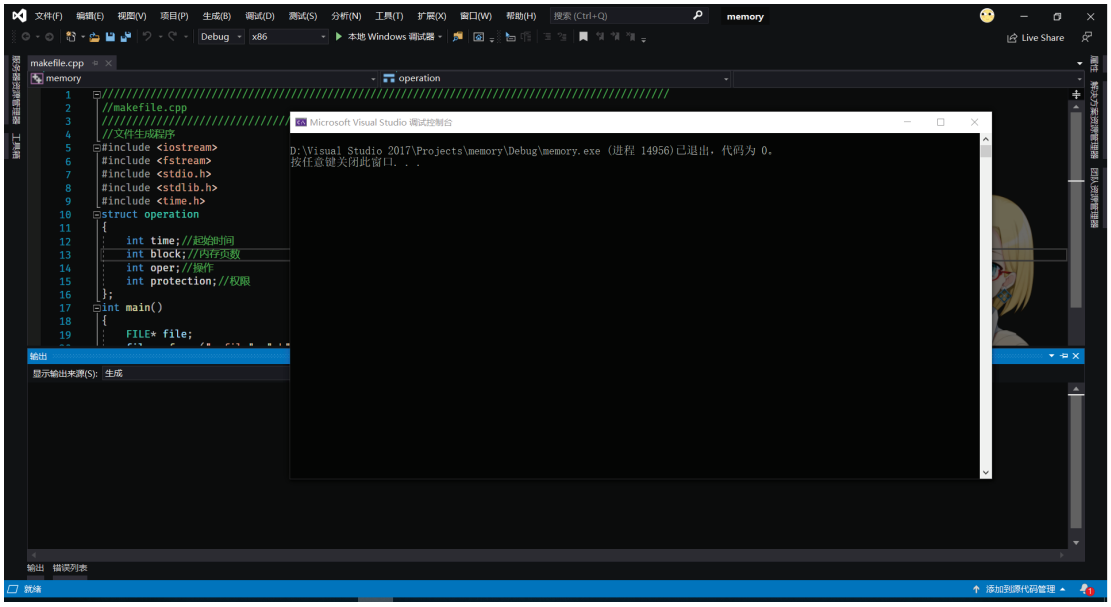
内存映射文件方法，适合于大型数据流文件以及多个进程之间的数据共享；

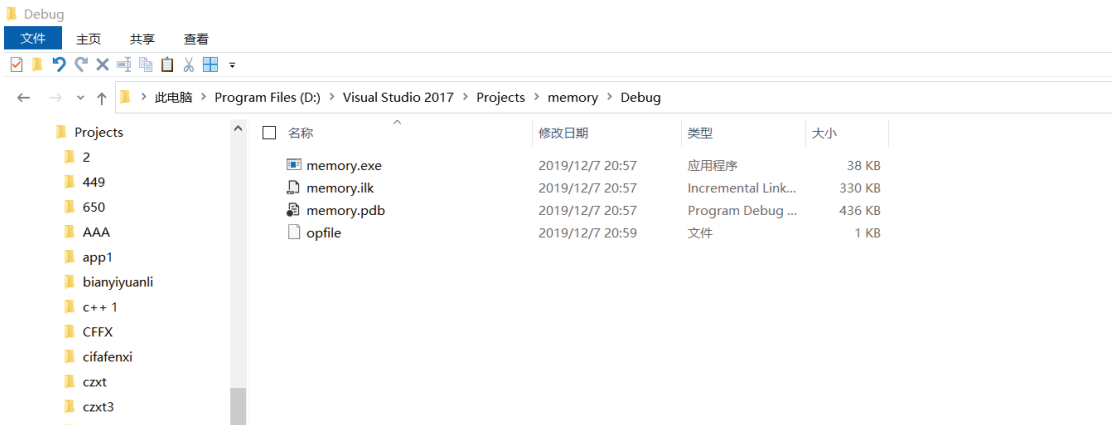
内存堆方法，适合于大量的小型内存申请。

本次实验主要是针对第一种使用方式。应用程序通过 API 函数 VirtualAlloc 和 VirtualAllocEx 等实现以页为单位的虚拟内存分配方法。首先保留地址空间，然后向此地址空间提交物理页面，也可以同时实现保留和提交。保留地址空间是为线程将来使用保留一块虚拟地址。在已保留的区域中，提交页面必须指出将物理存储器提交到何处以及提交多少。提交页面在访问时会转变为物理内存中的有效页面。

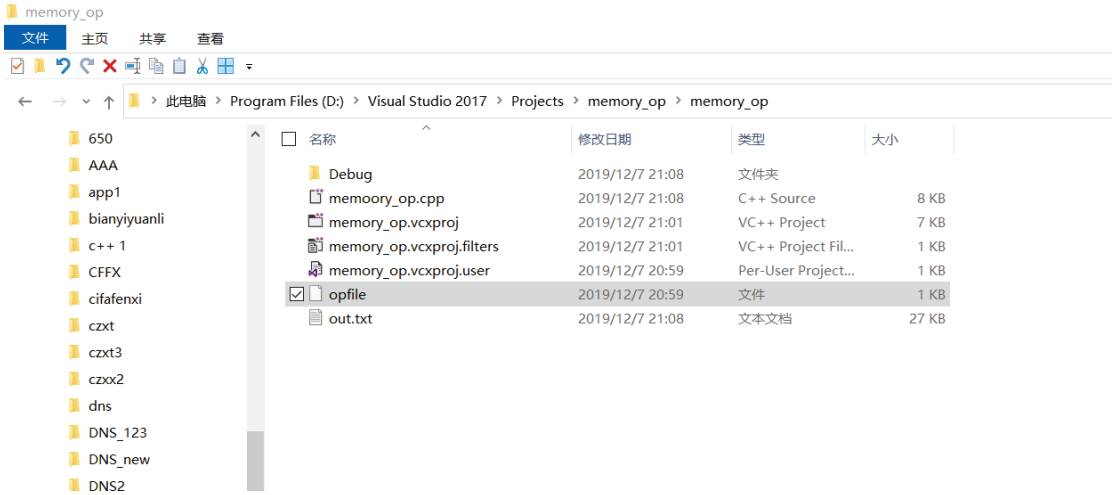
三、实验步骤：

1、编译运行 makefile.cpp，生成 opfile 文件。opfile 里记录了几个不同的内存操作，包括时间、块数、操作、大小、访问权限。生成了包括 5 个权限和 6 个操作任意组合共 30 个操作。



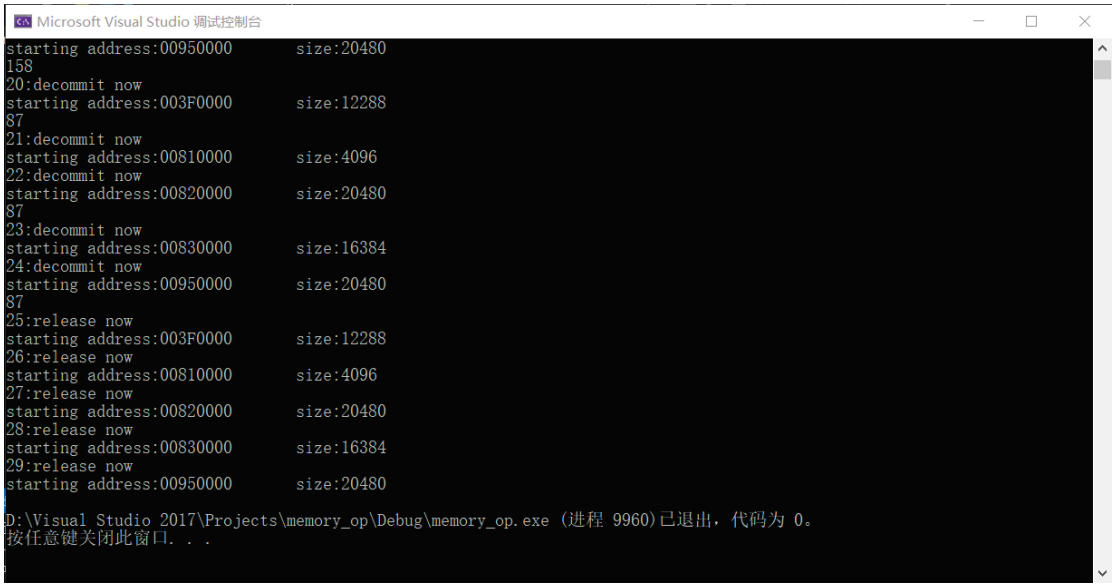


2、将生成的 opfile 复制到 memory-op 的工程目录下

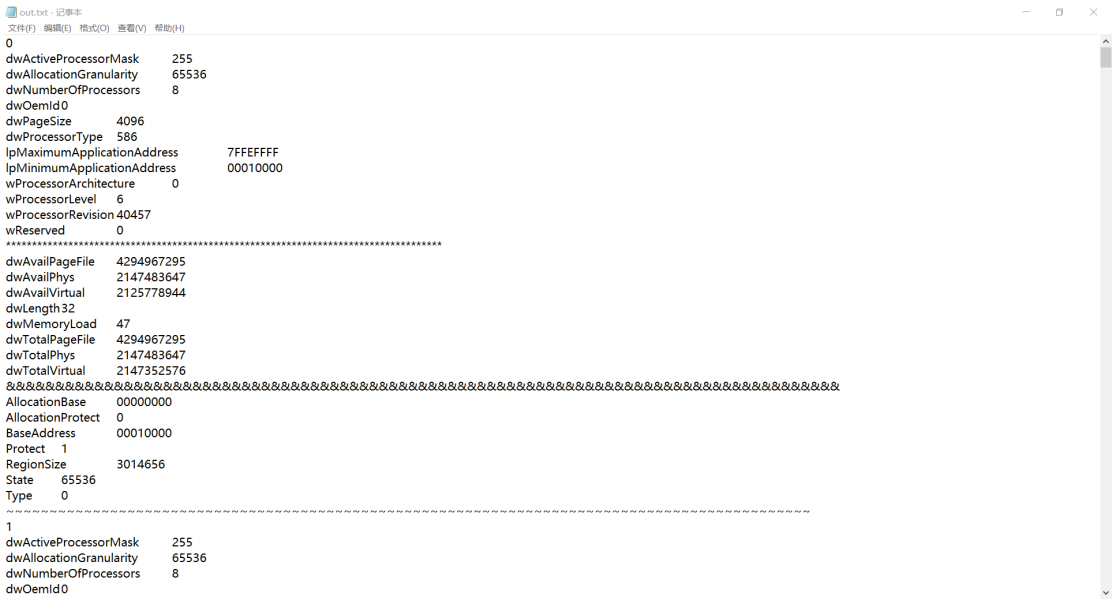


3、编译运行 memory-op.cpp，产生两个线程，一个从 opfile 文件里读取内存操作，模拟内存活动，另一个跟踪第一个的内存行为，将结果输出，并保存在 out.txt 文件中。

……



out.txt：



五、详细代码：

各函数的作用及详细实现和代码见附录工程文件。

六、相关的 API 函数

可以通过 GetSystemInfo，GlobalMemoryStatus 和 VirtualQuery 来查询进程虚空间的状态。主要的信息来源如下：

VOID GetSystemInfo （ LPSYSTEM_INFO lpSystemInfo ）；

结构 SYSTEMINFO 定义如下：

typedef struct _SYSTEM_INFO {

DWORD dwOemId;

DWORD dwPageSize;

LPVOID lpMinimumApplicationAddress;

LPVOID lpMaximumApplicationAddress;

DWORD dwActiveProcessorMask;
DWORD dwNumberOfProcessors;
DWORD dwProcessorType;
DWORD dwAllocationGranularity;
DWORD dwReserved;
} SYSTEM_INFO, *LPSYSTEM_INFO;
函数 VOID GlobalMemoryStatus （LPMEMORYSTATUS lpBuffer）;
数据结构 MEMORYSTATUS 定义如下：
typedef struct _ MEMORYSTATUS {
DWORD dwLength;
DWORD dwMemoryLoad;
DWORD dwTotalPhys;
DWORD dwAvailPhys;
DWORD dwTotalPageFile;
DWORD dwAvailPageFile;
DWORD dwTotalVirtual;
DWORD dwAvailVirtual;
} MEMORYSTATUS, * LPMEMORYSTATUS;
函数 DWORD VirtualQuery （ LPCVOLD lpAddress,
PMEMORY_BASIC_INFORMATION lpBuffer, DWORD dwLength）;
主要数据结构 MEMORY_BASIC_INFORMATION 定义如下：
typedef struct _ MEMORY_BASIC_INFORMATION {
PVOID BaseAddress;
PVOID AllocationBase;
DWORD AllocationProtect;
DWORD RegionSize;
DWORD State;
DWORD Protect;
DWORD Type;
} MEMORY_BASIC_INFORMATION;
typedef MEMORY_BASIC_INFORMATION * PMEMORY_BASIC_INFORMATION;
还有一些函数，例如 VirtualAlloc，VirtualAllocEx，VirtualFree 和 VirtualFreeEx 等，用
于虚拟内存的管理，详情请见 Microsoft 的 Win32 API Reference Manual。

七、实验总结：

通过内存管理实验，我深入地了解了 Windows 中的虚拟内存分配方式，熟悉了各种 Windows 下虚拟内存分配相关的 API 接口及函数的使用，让我受益良多。