北京理工大學

实验四 内存监视器

班　　级：07111505

姓　　名：徐宇恒

学　　号：1120151839

目录

一. 实验目的

熟悉 Windows 的存储器管理提供的各种机制，了解 Windows 的内存结构和虚拟内存的管理，学习如何在应用程序中管理内存

二. 实验内容

设计一个内存监视器，实现以下功能：

- 能实时的显示当前系统中的内存使用情况，包括系统地址空间的布局，物理内存的使用情况。
- 能实时显示某个进程的虚拟地址空间的布局和工作集信息

三. 实验环境

处理器	Inter(R) Core(TM) i7-6500U CPU
内存	8G
操作系统	Windows10 Pro 64bit

四. 实验过程

4.1. 基本思路

- 使用系统 API 函数 GetSystemInfo()获得系统的基本信息，包括内存的使用情况，系统的地址空间布局和物理内存的使用情况等。
- 使用进程管理 API 函数 CreateToolhelp32Snapshot()枚举系统中存在的进程，从而获得任意进程的名字，PID 等相关信息。
- 通过 OpenProcess()函数和 PID 得到进程的句柄
- 使用虚拟内存处理的 API 喊叔叔 VirtualQueryEx()和之前步骤得到的信息遍历指定进程的空间地址。

4.2. Windows 调用系统 API

4.2.1. GetSystemInfo()

```
SYSTEM_INFO si;
ZeroMemory(&si, sizeof(si));
GetSystemInfo(&si);

SYSTEM_INFO 的结构如下

typedef struct _SYSTEM_INFO {
     union {
          DWORD dwOemId; // Obsolete field...do not use
          struct {
               WORD wProcessorArchitecture;
               WORD wReserved;
          } DUMMYSTRUCTNAME;
     } DUMMYUNIONNAME;
     DWORD dwPageSize; // 内存页的大小
     LPVOID lpMinimumApplicationAddress; // 每个进程可用的地址空间最小内存地址
     LPVOID lpMaximumApplicationAddress; // 每个进程可用的地址空间最大内存地址
     DWORD_PTR dwActiveProcessorMask;
     DWORD dwNumberOfProcessors;
     DWORD dwProcessorType;
     DWORD dwAllocationGranularity; // 能够保留地址空间区域的最小单位，win32下默认为
64KB
     WORD wProcessorLevel;
     WORD wProcessorRevision;
}SYSTEM_INFO,*LPSYSTEM_INFO;
```

4.2.2. CreateToolhelp32Snapshot()

- 通过获取进程信息为指定的进程、进程使用的堆、模块、线程间里一个快照

4.2.3. OpenProcess()

打开进程，获取句柄

```
HANDLE hProcess = OpenProcess(
     PROCESS_QUERY_INFORMATION | PROCESS_VM_READ, // 欲进行的操作
     FALSE, // 继承属性
     processID); // 欲查询的进程PID
```

4.2.4. VirtualQueryEx()

查询地址空间中内存地址信息

```
SIZE_T WINAPI VirtualQueryEx(
     _In_ HANDLE hProcess, //进程句柄
     _In_opt_ LPCVOID lpAddress, //查询内存的地址
     _Out_ PMEMORY_BASIC_INFORMATION lpBuffer, //指向MEMORY_BASIC_INFORMATION结构
的指针，用于接收内存信息
```

```
      _In_ SIZE_T dwLength //MEMORY_BASIC_INFORMATION结构的大小
);
```

4.2.5. GlobalMemoryStatus()

获得当前可用物理地址和虚拟内存信息

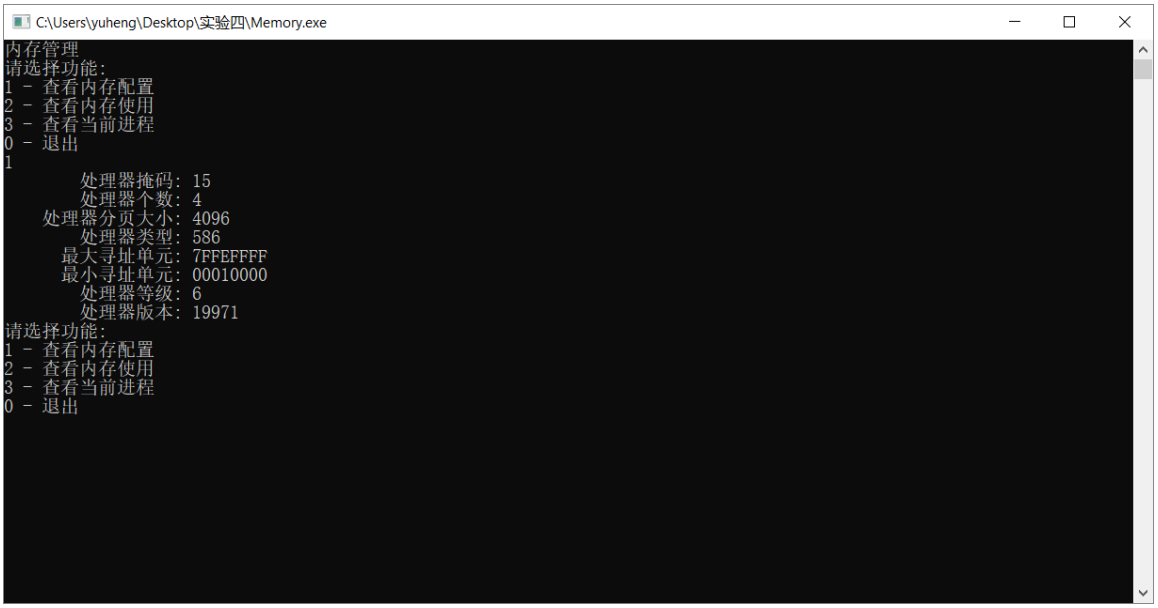
```
VOID GlobalMemoryStatus
(
      LPMEMORYSTATUS lpBuffer
);
```

其中 MEMORYSTATUS 的结构如下

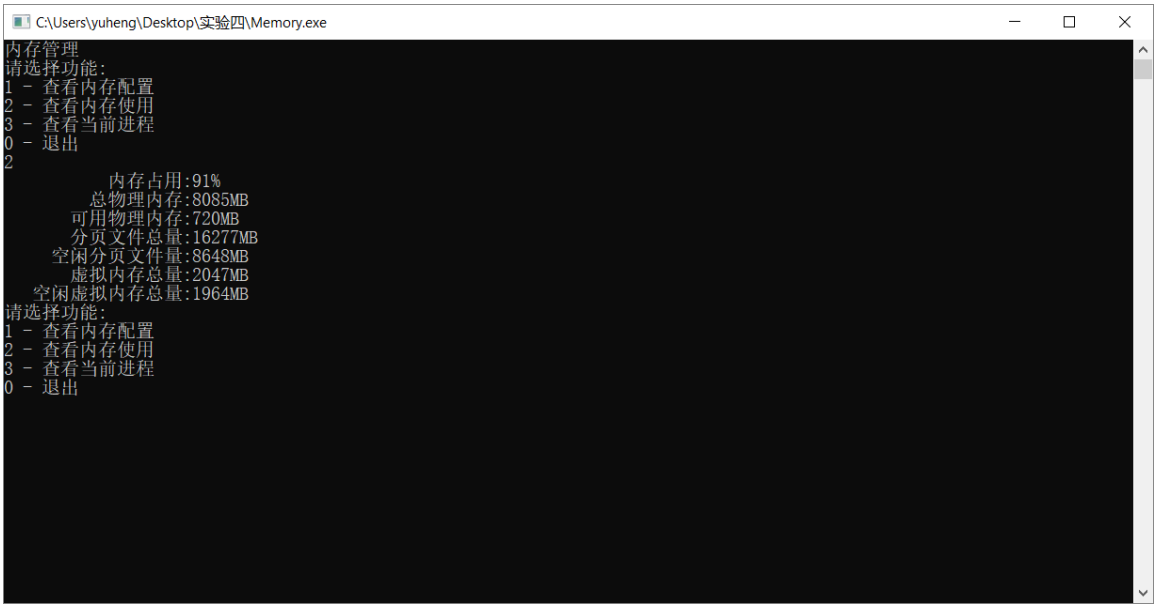
```
typedef struct _MEMORYSTATUS { // mst
      DWORD dwLength; // sizeof(MEMORYSTATUS)
      DWORD dwMemoryLoad; // percent of memory in use
      DWORD dwTotalPhys; // bytes of physical memory
      DWORD dwAvailPhys; // free physical memory bytes
      DWORD dwTotalPageFile; // bytes of paging file
      DWORD dwAvailPageFile; // free bytes of paging file
      DWORD dwTotalVirtual; // user bytes of address space
      DWORD dwAvailVirtual; // free user bytes
} MEMORYSTATUS, *LPMEMORYSTATUS;
```

五. 实验结果

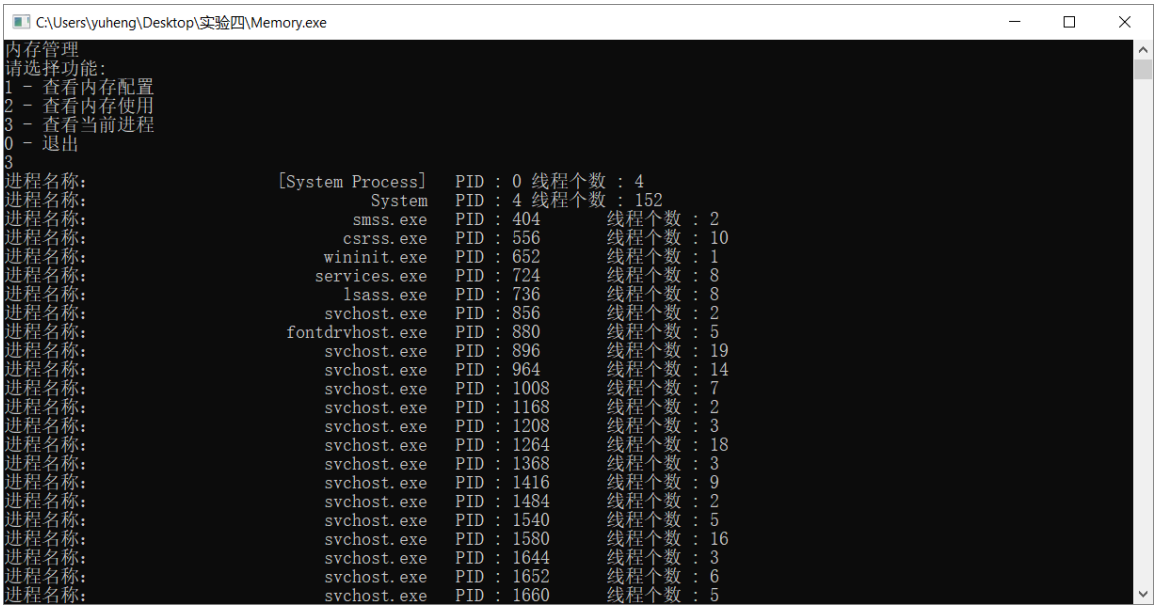
按 1 查看内存配置



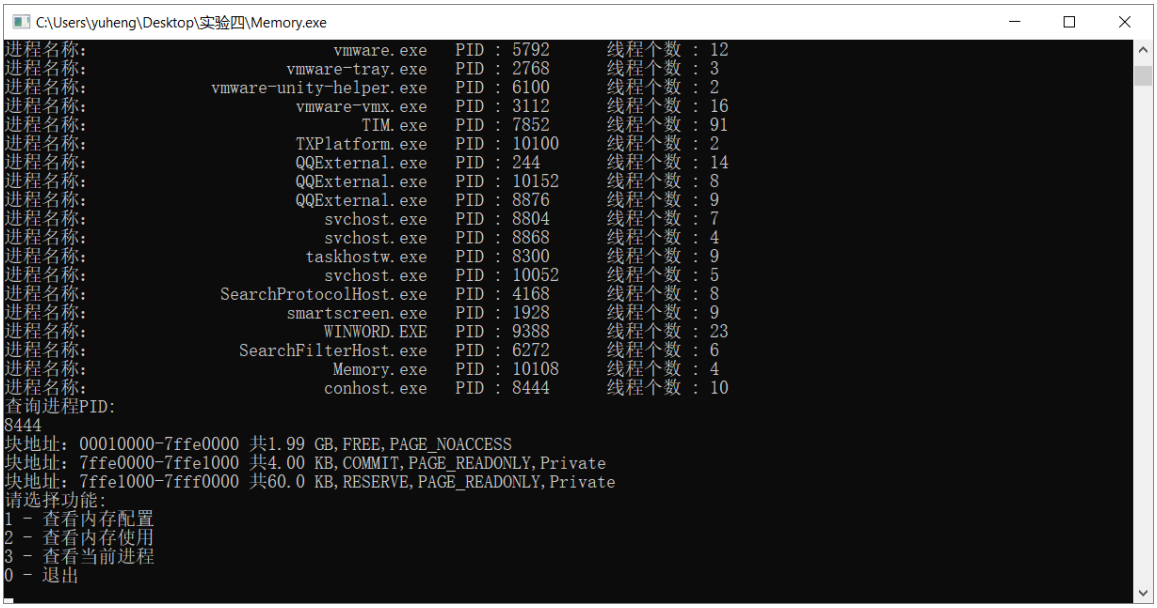
按 2 查看内存使用情况



按 3 查看当前正在运行的进程快照



输入进程 PID 查看详细使用情况



六. 心得体会

实验四重要是调用 Windows 已经封装好的 API 函数，只要明白其调用的方法以及返回值的类型即可完成。操作简单快捷，使我了解到了如何通过查阅 Windows 的文档来正确使用这些已有的工具。

而且在实验的过程中，我也切实了解到了 Windows 进成地址空间，如何利用操作系统的虚拟内存机制来增强程序对内存的管理能力。