实验三 遍历进程地址空间

班级： 07112004 学号： 1120201883 姓名： 刘博文

## 一、实验目的

学习怎么在windows平台上查看进程的空间等信息

## 二、实验内容

能实时显示某个进程的虚拟地址空间布局和工作集信息等。

在Windows平台上做。相关的系统调用：

GetSystemInfo, VirtualQueryEx,VirtualAlloc, GetPerformanceInfo, GlobalMemoryStatusEx …

## 三、程序设计与实现

首先要看一下要用到的函数以及结构体：

**结构体SYSTEM\_INFO**

就和名字一样，这个结构体描述了系统的一些基本信息，重要的信息用注释标注了出来：

typedef struct \_SYSTEM\_INFO {  
union {  
  DWORD dwOemId;  
  struct {  
    WORD wProcessorArchitecture;  
    WORD wReserved;  
  } DUMMYSTRUCTNAME;  
} DUMMYUNIONNAME;  
//页面大小以及页面保护和承诺的粒度。这是VirtualAlloc函数使用的页面大小 。  
DWORD     dwPageSize;  
//可访问的最低内存地址的指针。     
LPVOID    lpMinimumApplicationAddress;  
//可访问的最高内存地址的指针。  
LPVOID    lpMaximumApplicationAddress;  
DWORD\_PTR dwActiveProcessorMask;  
//逻辑处理器的数量。  
DWORD     dwNumberOfProcessors;  
DWORD     dwProcessorType;  
DWORD     dwAllocationGranularity;  
WORD      wProcessorLevel;  
WORD      wProcessorRevision;  
} SYSTEM\_INFO, \*LPSYSTEM\_INFO;

**函数GetSystemInfo**

WINBASEAPI VOID WINAPI GetSystemInfo (LPSYSTEM\_INFO lpSystemInfo);

lpSystemInfo指向接收信息的SYSTEM\_INFO结构的指针 ，系统的信息会返回到参数的指针指向的SYSTEM\_INFO类型的结构体

**结构体PERFORMANCE\_INFORMATION**

typedef struct \_PERFORMANCE\_INFORMATION {  
//结构体大小  
DWORD  cb;  
   //系统当前提交的页数  
SIZE\_T CommitTotal;  
   //系统可以提交的最大页数  
SIZE\_T CommitLimit;  
   //系统自上次重启以来的处于已提交状态的最大页数  
SIZE\_T CommitPeak;  
SIZE\_T PhysicalTotal;  
SIZE\_T PhysicalAvailable;  
   //以页为单位的系统缓存内存量。 这是备用列表的大小以及系统工作集。  
SIZE\_T SystemCache;  
   //分页和非分页内核池中当前内存的总和（以页为单位）  
SIZE\_T KernelTotal;  
SIZE\_T KernelPaged;  
SIZE\_T KernelNonpaged;  
   //页大小  
SIZE\_T PageSize;  
   //总HANDLE数  
DWORD  HandleCount;  
   //总进程数  
DWORD  ProcessCount;  
   //总线程数  
DWORD  ThreadCount;  
} PERFORMANCE\_INFORMATION, \*PPERFORMANCE\_INFORMATION, PERFORMACE\_INFORMATION, \*PPERFORMACE\_INFORMATION;

**函数GetPerformanceInfo**

WINBOOL WINAPI GetPerformanceInfo (PPERFORMACE\_INFORMATION pPerformanceInformation,DWORD cb);

将内存性能信息返回到第一个参数中

**结构体MEMORYSTATUSEX**

typedef struct \_MEMORYSTATUSEX  
{  
   //本结构长度  
DWORD     dwLength;  
   //已用内存百分比  
DWORD     dwMemoryLoad;   
   //物理内存总量  
DWORDLONG    ullTotalPhys;   
   //可用物理内存  
DWORDLONG    ullAvailPhys;   
   //页交换文件总大小  
DWORDLONG    ullTotalPageFile;  
   //页交换文件可用大小  
DWORDLONG    ullAvailPageFile;  
   //用户区总的虚拟地址空间  
DWORDLONG    ullTotalVirtual;   
   //用户区当前可用的虚拟地址空间  
DWORDLONG    ullAvailVirtual;   
DWORDLONG    ullAvailExtendedVirtual;  
} MEMORYSTATUSEX,  \*MEMORYSTATUSEX;

**函数GlobalMemoryStatusEx**

WINBASEAPI WINBOOL WINAPI GlobalMemoryStatusEx (LPMEMORYSTATUSEX lpBuffer);

参数是一个指向一个MEMORYSTATUSEX结构的指针，函数会将系统当前的全局内存信息写入到该指针所指向的MEMORYSTATUSEX结构，并返回一个bool值来确定该次内存信息获取是否成功

注意：lpBuffer指向的结构体的dwLength字段必须在调用该函数前赋值为该结构体的长度，不然该函数不会返回正确结果

**结构体PROCESSENTRY32**

typedef struct tagPROCESSENTRY32 {  
   //结构体大小，使用之前也要赋值  
   DWORD dwSize;  
   DWORD cntUsage;  
   //进程唯一的PID  
   DWORD th32ProcessID;  
   ULONG\_PTR th32DefaultHeapID;  
   DWORD th32ModuleID;  
   //进程启动的执行线程数  
   DWORD cntThreads;  
   DWORD th32ParentProcessID;  
   LONG pcPriClassBase;  
   DWORD dwFlags;  
   //进程文件的名称  
   CHAR szExeFile[MAX\_PATH];  
} PROCESSENTRY32;

**函数CreateToolhelp32Snapshot**

HANDLE WINAPI CreateToolhelp32Snapshot(DWORD dwFlags,DWORD th32ProcessID);

拍摄指定进程以及这些进程使用的堆，模块和线程的快照，返回一个HANDLE

**函数Process32First Process32Next**

WINBOOL WINAPI Process32First(HANDLE hSnapshot,LPPROCESSENTRY32 lppe);  
WINBOOL WINAPI Process32Next(HANDLE hSnapshot,LPPROCESSENTRY32 lppe);

函数Process32First的作用是检索快照hSnapshot中第一个进程的信息，并把这些信息存储在了指针lppe所指向的结构体中。检索成功则返回true，否则返回false

函数Process32Next的作用是检索快照hSnapshot中第下一个进程的信息，并把这些信息存储在了指针lppe所指向的结构体中。检索成功则返回true，否则返回false

**函数OpenProcess**

WINBASEAPI HANDLE WINAPI OpenProcess (DWORD dwDesiredAccess, WINBOOL bInheritHandle, DWORD dwProcessId);

该函数的作用是打开一个本地进程对象。第一个参数是访问权限，第二个是进程是否继承句柄，第三个是要打开的进程的PID

若函数成功，则返回指定进程的打开HANDLE；若失败则返回NULL

**结构体MEMORY\_BASIC\_INFORMATION**

//存储页面信息  
typedef struct \_MEMORY\_BASIC\_INFORMATION {  
   //页面基地址  
   PVOID BaseAddress;  
   PVOID AllocationBase;  
   DWORD AllocationProtect;  
   //区域大小  
   SIZE\_T RegionSize;  
   //页面状态  
DWORD State;  
   //页面保护权限  
DWORD Protect;  
   //页面类型  
DWORD Type;  
} MEMORY\_BASIC\_INFORMATION,\*PMEMORY\_BASIC\_INFORMATION;

**VirtualQueryEx**

WINBASEAPI SIZE\_T WINAPI VirtualQueryEx (HANDLE hProcess, LPCVOID lpAddress, PMEMORY\_BASIC\_INFORMATION lpBuffer, SIZE\_T dwLength);

检索有关指定进程的虚拟地址空间内的页面范围的信息

### 程序设计思路

#### 1.系统信息

使用SYSTEM\_INFO保存系统信息，使用函数GetSystemInfo获取系统信

息，然后按照SYSTEM\_INFO中的字段输出打印系统信息，包括最小地址、最高地址和页大小等

#### 2.内存性能信息

使用PERFORMANCE\_INFORMATION结构保存函数GetPerformanceInfo获取到

的内存性能信息，并将其输出，包括总提交页数、最大提交页数、系统缓存页数、内核池中的页数总和、页大小、HANDLE数量、总进程数、总线程数等

#### 3.全局内存信息

使用MEMORYSTATUSEX结构保存函数GlobalMemoryStatusEx所获取到的内存状态，再将相关信息打印输出，包括物理内存总量、可用物理内存总量、内存占用率、总虚拟内存、可用虚拟内存、系统或当前进程的当前已提交内存限制、当前进程可以提交的最大内存量等

#### 4.全局进程信息

使用函数CreateToolhelp32Snapshot获取指定进程的快照，以及这些进程使用的堆、模块和线程。注意该函数的第一个参数按照微软官方的文档应该是”TH32CS\_SNAPPROCESS”，这样才能在快照中包含所有进程。

之后使用函数Process32First和Process32Next获取快照中保存的所有进程的PID并将其打印输出，方便后面选择具体的进程获取其信息

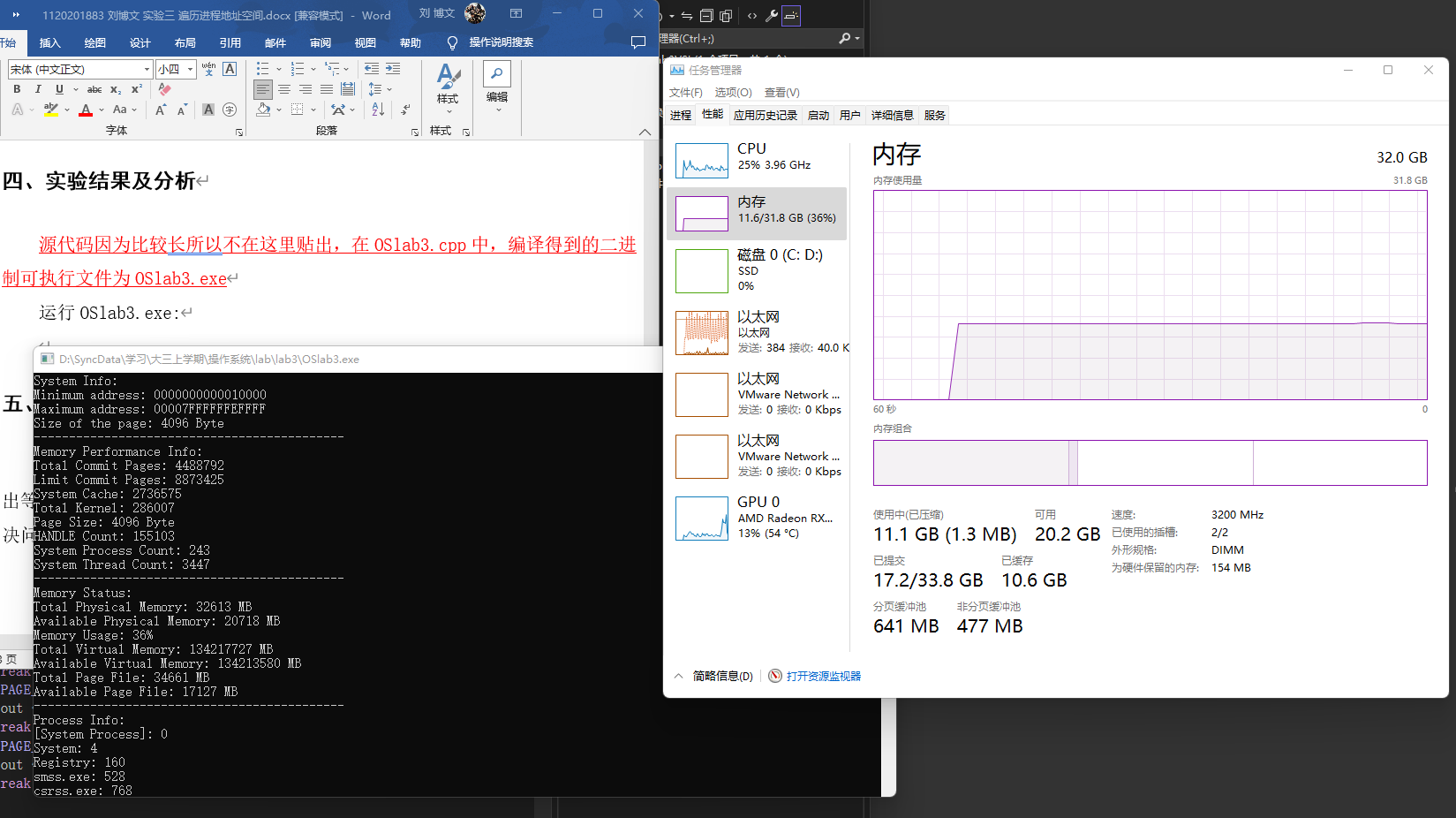
#### 5.查询某特定进程信息

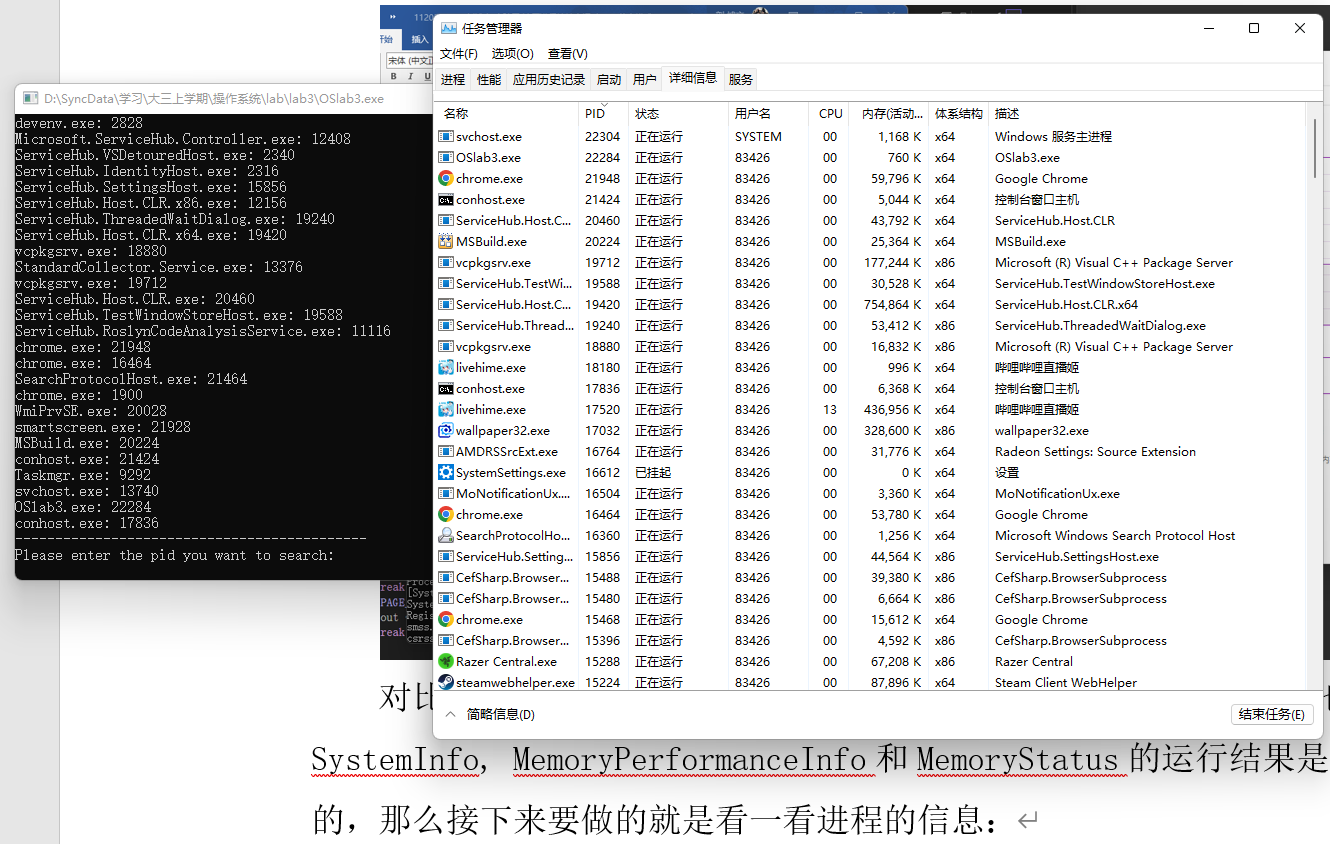
用户输入某进程的PID（若输入的PID不存在与之对应的进程则会提示用户重新输入直至正确为止），然后根据PID使用函数OpenProcess获取该进程的打开HANDLE。

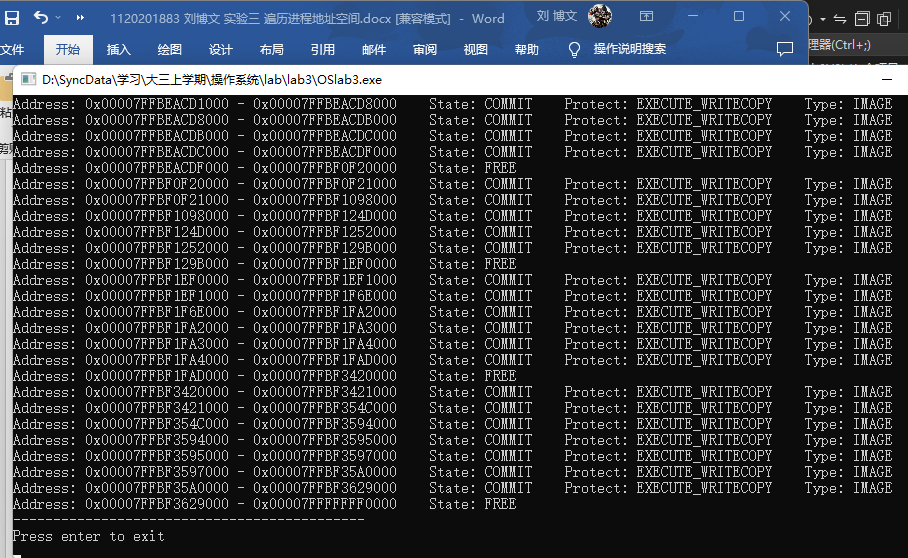
之后用结构MEMORY\_BASIC\_INFORMATION保存函数VirtualQueryEx获取到的进程虚拟地址空间中的页面范围信息，并将其输出

## 四、实验结果及分析

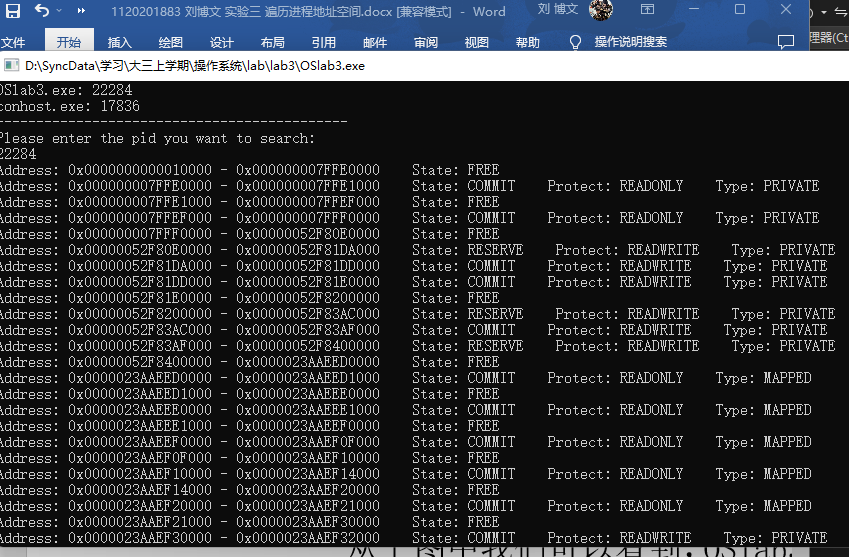
源代码因为比较长所以不在这里贴出，在OSlab3.cpp中，编译得到的二进制可执行文件为OSlab3.exe

运行OSlab3.exe:

对比程序运行内容与任务管理器显示的数据，可以发现前三部分也就是SystemInfo, MemoryPerformanceInfo和MemoryStatus的运行结果是符合预期的，那么接下来要做的就是看一看进程的信息：

从上图中我们可以看到，OSlab3.exe我们显示的PID与任务管理器中的PID都是22284，chrome.exe都是21948等。因为进程太多了，不可能一个一个的去确认，但是基本可以确定这没什么问题。接下来输入具体的PID查看信息：

在这里我查看的是OSlab3.exe的进程信息，因为这一块我不知道怎么验证结果的正确性，但是功能是没有问题的.

最后输入enter即可退出程序:

## 五、实验收获与体会

做这个实验真的挺麻烦的，遇到了乱码、使用cout无输出但是printf有输出等奇奇怪怪的现象，而且一些winAPI真的挺麻烦的，不过好在慢慢学习、解决问题之后也完成了这个实验