1. **实验内容及目的**

设计一个内存监视器，能实时地显示当前系统中内存的使用情况，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布局和工作集信息等（通过PID查询进程虚拟地址空间情况）。

**二、实验环境**

操作系统：windows10 64位

编译器：Visual Studio 2012

**三、基本原理及方法概述**

**3.1用到的系统调用函数**

**(1)GetSystemInfo()函数**

获得当前系统的一些特征信息

Void WINAPI GetSystemInfo(LPSYSTEM\_INFO lpSystemInfo);

//lpSystemInfo为指向SYSTEM\_INFO结构体的指针。

**(2)StrFormatByteSize()函数**

将数字转换成字符串

LPTSTR StrFormatBytesize(

LONGLONG qdw, //要转变的数字值

LPTSTR pszBuf, //指向保存将数字转变成字符串的缓冲区指针

UNIT uiBufSize //缓冲区的容量

);

1. **VirtualQueryEx()函数**

检查进程虚拟内存的当前信息

SIZE\_T WINAPI VirtualQueryEx(

HANDLE hProcess,//进程句柄

LPCVOID lpAddress,//指向要查询的页基地址指针

PMEMORY\_BASIC\_INFORMATION lpBuffer,//指向 MEMORY\_BASIC\_INFORMATION结构的缓冲区指针，用来接收要查询的内存信息

SIZE\_T dwLength //MEMORY\_BASIC\_INFORMATION结构的大小

);

**(4)GetPerformanceInfo()函数**

获取系统的性能信息

BOOL WINAPI GetPerformanceInfo(

PPERFORMANCE\_INFORMATION pPerformanceInformation,

//为指向PERFORMANCE\_INFORMATION结构体的指针，该结构体用来存储系统性能信息。

DWORD cb //PERFORMANCE\_INFORMATION结构体的大小

);

**(5)GlobalMemoryStatusEx()函数**

获取系统当前物理内存和虚拟内存的使用情况

BOOL WINAPI GlobalMemoryStatusEx(LPMEMORYSTATUSEXlpBuffer);

//lpBuffer为指向MEMORYSTATUSEX结构体的指针

## **CreateToolhelp32Snapshot()函数**

通过获取进程信息为指定的进程、进程使用的堆[HEAP]、模块[MODULE]、[线程](https://baike.baidu.com/item/%E7%BA%BF%E7%A8%8B" \t "https://baike.baidu.com/item/CreateToolhelp32Snapshot/_blank)建立一个快照,可以获取系统中正在运行的进程信息、线程信息等。用于通过指定的pid查询进程的工作集和地址空间布局。

HANDLE WINAPI CreateToolhelp32Snapshot(

DWORD dwFlags, //用来指定“快照”中需要返回的对象，可以是TH32CS\_SNAPPROCESS等

DWORD th32ProcessID //一个进程ID号，用来指定要获取哪一个进程的快照，当获取系统进程列表或获取 当前进程快照时可以设为0

);

## **process32Firs()函数**

process32Firs是一个进程获取函数，当我们利用函数CreateToolhelp32Snapshot()获得当前运行进程的快照后，我们可以利用process32First函数来获得第一个进程的句柄。

## **GetProcessMemoryInfo()函数**

用于获取进程内存的使用情况

BOOL WINAPI GetProcessMemoryInfo(

HANDLE　Process, //获取内存使用情况的进程句柄

PPROCESS\_MEMORY\_COUNTERS　ppsmemCounters, //返回内存使用情况的结构体

DWORD　cb //结构体的大小

);

**3.2用到的结构体**

## **(1)SYSTEM\_INFO结构体**

记录当前系统的一些特征信息的结构体

SYSTEM\_INFO

typedef struct \_SYSTEM\_INFO

{

union{

DWORD dwOemId;

struct{

WORD wProcessorArchitecture;

WORD wReserved;

};

};

DWORD dwPageSize;//内存页的大小

LPVOID lpMinimumApplicationAddress;//进程地址空间的最小内存地址

LPVOID lpMaximumApplicationAddress;//进程地址空间的最大内存地址

DWORD\_PTR dwActiveProcessorMask;//系统配备的CPU的掩码

DWORD dwNumberOfProcessors;//系统配备的CPU的数量

DWORD dwProcessorType;//系统配备的CPU的类型，向下兼容用

DWORDdwAllocationGranularity;//能够保留地址空间区域的最小单位，win32

默认64KB

WORD wProcessorLevel;//CPU的级别

WORD wProcessorRevision;//步进级别

}SYSTEM\_INFO;

**(2)MEMORYSTATUSEX结构体**

记录系统物理内存和虚拟内存信息的结构体

MEMORYSTATUSEX

typedef struct \_MEMORYSTATUSEX

{

DWORD dwLength;//结构体的大小

DWORD dwMemoryLoad;//物理内存使用率

DWORDLONG ullTotalPhys;//总的物理内存

DWORDLONG ullAvailPhys;//可用物理内存

DWORDLONG ullTotalPageFile;//总的交换文件

DWORDLONG ullAvailPageFile;//可用的交换文件

DWORDLONG ullTotalVirtual;//总的虚拟内存

DWORDLONG ullAvailVirtual;//可用的虚拟内存

DWORDLONG ullAvailExtendedVirtual;//扩展内存

}MEMORYSTATUSEX,\*LPMEMORYSTATUSEX;

**(3)PERFORMANCE\_INFORMATION结构体**

记录系统性能信息的结构体

PERFORMANCE\_INFORMATION

typedefstruct\_PERFORMANCE\_INFORMATION

{

DWORD cb;//结构体大小（按字节来算）

SIZE\_T CommitTotal;//系统当前提交的页面总数

SIZE\_T CommitLimit;//系统当前可提交的最大页面总数

SIZE\_T CommitPeak;//系统历史提交页面峰值

SIZE\_T PhysicalTotal;//按页分配的总物理内存

SIZE\_T PhysicalAvailable;//当前可用的物理内存

SIZE\_T SystemCache;//系统cache容量

SIZE\_T KernelTotal;//内存总量（按页）

SIZE\_T KernelPaged;//分页池大小

SIZE\_T KernelNonpaged;//非分页池大小

SIZE\_T PageSize;//页大小

DWORD HandleCount;//打开的句柄个数

DWORD ProcessCount;//进程个数

DWORD ThreadCount;//线程个数

}PERFORMANCE\_INFORMATION,\*PPERFORMANCE\_INFORMATION;

1. PROCESS\_MEMORY\_COUNTERS结构体

存储进程的工作集信息

typedef struct \_PROCESS\_MEMORY\_COUNTERS {

DWORD cb;

DWORD PageFaultCount; //缺页中断次数

SIZE\_T PeakWorkingSetSize;//使用内存的高峰

SIZE\_T WorkingSetSize;//当前使用的内存

SIZE\_T QuotaPeakPagedPoolUsage;//使用页面缓冲池高峰

SIZE\_T QuotaPagedPoolUsage;//当前使用页面缓冲池

SIZE\_T QuotaPeakNonPagedPoolUsage;//使用非分页缓冲池高峰

SIZE\_T QuotaNonPagedPoolUsage;//使用非分页缓冲池

SIZE\_T PagefileUsage;//使用分页文件

[SIZE](https://baike.baidu.com/item/SIZE" \t "https://baike.baidu.com/item/PROCESS_MEMORY_COUNTERS/_blank)\_T PeakPagefileUsage;//使用分页文件高峰

} PROCESS\_MEMORY\_COUNTERS, \*PPROCESS\_MEMORY\_COUNTERS;

**(5)MEMORY\_BASIC\_INFORMATION结构体**

接收要查询的内存信息结构体

MEMORY\_BASIC\_INFORMATION

typedef struct \_MEMORY\_BASIC\_INFORMATION

{

PVOID BaseAddress;//基地址的最小页号(按页对齐方式分配时)

PVOID AllocationBase;//应用程序的实际起始地址

DWORD AllocationProtect;//区域初始设置的访问方式

SIZE\_T RegionSize;//虚存区的大小

DWORD State;//该区域的状态

DWORD Protect;//该区域设置的访问方式

DWORD Type;//页面类型

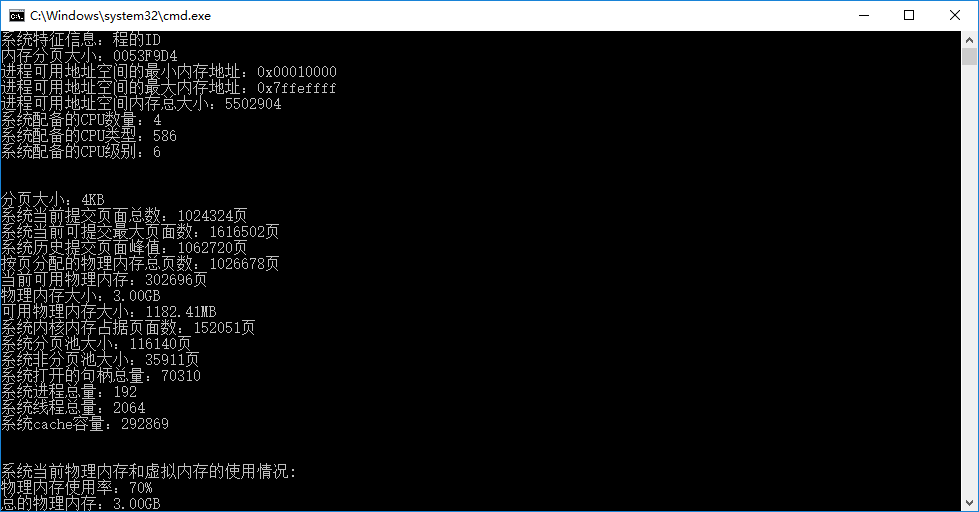
}MEMORY\_BASIC\_INFORMATION,\*PMEMORY\_BASIC\_INFORMATION;

**3.3其它函数**

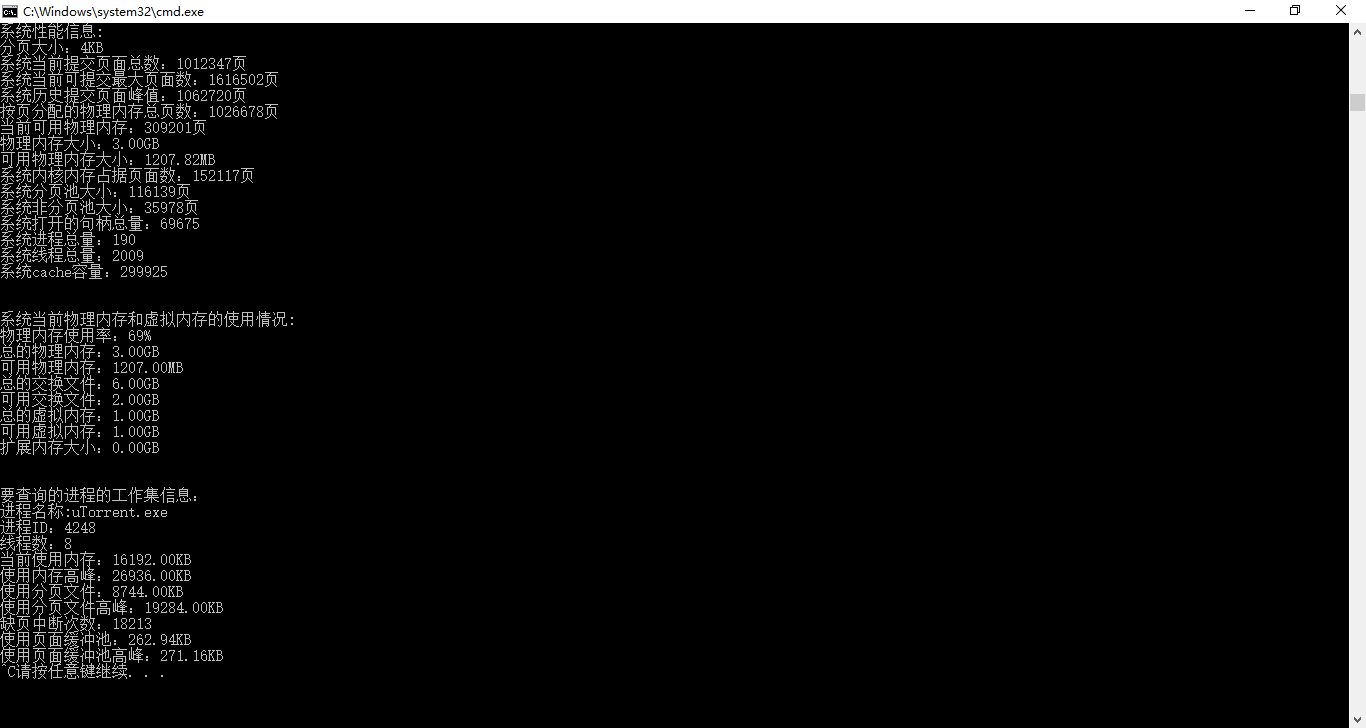
在获取进程虚拟地址空间信息时采用了《操作系统实验教程及Linux和windows系统调用编程》张丽芬 刘昕等编著一书中P291页利用VirtualQueryEx()函数检测进程的虚拟地址空间信息的示例代码中的WalkVM(HANDLE hProcee)函数。通过传入此函数一个进程的句柄即可输出进程的虚拟地址空间布局。

1. **实验结果截图及分析**

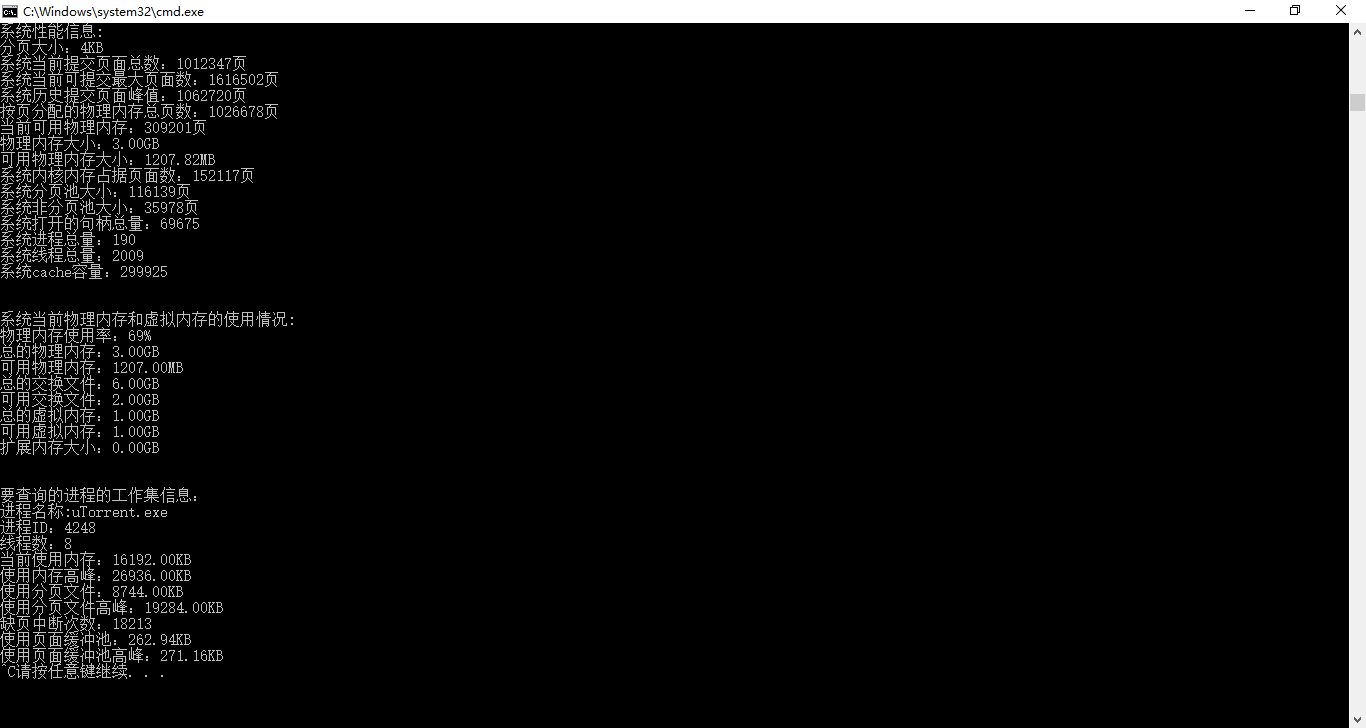
**4.1利用GetSystemInfo()函数获取的系统特征信息**



**4.2利用GlobalMemoryStatusEx()函数获取的系统当前物理内存和虚拟内存的使用情况**



**4.3利用GetPerformanceInfo()函数获取的系统性能信息**



**4.4实时显示指定进程的虚拟地址空间布局和工作集信息**

