操作系统课程设计 实验报告

姓名：刘天祺

班级：07121502

学号：1320151097

学院：计算机学院

专业：物联网工程

日期：2018 年 4 月 17 日

实验四　内存监视器

一、实验要求

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）。

仅实现 Windows 版本。

二、实验环境及配置

操作系统：Windows 10 64bit

Shell：cmder 160710

编译器：gcc 3.4.5 (mingw-vista special r3)

编译参数：-lpsapi

三、实验步骤

- 编写交互式程序入口

```
171  void usage(){
172      printf("\n");
173      printf("1.show SYSTEM Infomation\n");
174      printf("2.show MEMORY Infomation\n");
175      printf("3.show PROCESS Infomation\n");
176      printf("4.show target PROCESS Info\n");
177  }
178
179  int main() {
180      usage();
181      while (1){
182          printf("\n$ ");
183          int cmd = 0;
184          scanf("%d", &cmd);
185          switch(cmd){
186              case 1: showSys(); break;
187              case 2: showMem(); break;
188              case 189: showAllProc(); break;
189              case 4: showSingleProcess(); break;
190              default: exit(0);
191          }
192      }
193  }
```

用户可以输入命令 1-4，分别查询系统地址空间的布局、物理内存的使用情况、所有进程的信息、单个进程的信息及其虚拟地址空间布局。

- 显示系统地址空间的布局

```
6   void showSys(){
7       SYSTEM_INFO si;
8       ZeroMemory(&si, sizeof(si));
9       GetSystemInfo(&si);
10
11      LPCVOID *MinAddr = si.lpMinimumApplicationAddress;
12      LPCVOID *MaxAddr = si.lpMaximumApplicationAddress;
13
14      printf("Page Size: %d KB\n", si.dwPageSize / 1024);
15      printf("Processor Num: %d\n", si.dwNumberOfProcessors);
16      printf("Cpu Arch: %d\n", si.dwProcessorType);
17      printf("VM Fineness: %d KB\n", si.dwAllocationGranularity / 1024);
18      printf("Valiable VM: %0.2f GB\n",
19          ((DWORD)MaxAddr - (DWORD)MinAddr) / (1024.0*1024.0*1024.0));
20      printf("Range Of VM: 0x%08x - 0x%08x\n", MinAddr, MaxAddr);
21      return ;
22  }
23
```

通过调用 API GetSystemInfo()获取系统信息，并计算输出相应系统信息。

- 显示系统物理内存的使用情况

```
24  void showMem(){
25      MEMORYSTATUS ms;
26      ms.dwLength = sizeof(MEMORYSTATUS);
27      GlobalMemoryStatus(&ms);
28
29      printf("Memory Used: %d%%\n", ms.dwMemoryLoad);
30      DWORD TotalPhys = ms.dwTotalPhys;
31      DWORD AvailPhys = ms.dwAvailPhys;
32      printf("Phys Mem: %.2fM/%.2fM Free:%.1f%%\n",
33          (float)AvailPhys/1024/1024,
34          (float)TotalPhys/1024/1024,
35          (1-(float)AvailPhys/TotalPhys)*100);
36      DWORD TotalVirs = ms.dwTotalVirtual;
37      DWORD AvailVirs = ms.dwAvailVirtual;
38      printf("Virs Mem: %.2fM/%.2fM Free:%.1f%%\n",
39          (float)AvailVirs/1024/1024,
40          (float)TotalVirs/1024/1024,
41          (1-(float)AvailVirs/TotalVirs)*100);
42      return ;
43  }
```

通过调用 GlobalMemoryStatus()获取系统内存使用情况并计算、输出相应信息。

- 查询所有进程的信息

```
38  void showAllProc(){
39      // get the snapshoot
40      HANDLE hProcessSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
41      if( hProcessSnap==INVALID_HANDLE_VALUE ) {
42          printf("CreateToolhelp32Snapshot fialed\n");
43          return ;
44      }
45
46      // get info for each process
47      PROCESSENTRY32 stcProcessInfo;
48      stcProcessInfo.dwSize = sizeof(stcProcessInfo);
49
50      BOOL bRet = Process32First(hProcessSnap, &stcProcessInfo);
51      PROCESS_MEMORY_COUNTERS pmc;
52      pmc.cb = sizeof(pmc);
53
54      while (bRet) {
55          // process info
56          printf("Name:\t%s\nPID:\t%d\nThreads:%d\nPPID:\t%d\n",
57              stcProcessInfo.szExeFile,
58              stcProcessInfo.th32ProcessID,
59              stcProcessInfo.cntThreads,
60              stcProcessInfo.th32ParentProcessID);
61
62          // workset infomation
63          HANDLE hProcess =  OpenProcess(PROCESS_ALL_ACCESS, TRUE, stcProcessInfo.th32ProcessID);;
64
65          GetProcessMemoryInfo(hProcess, &pmc, sizeof(pmc));
66          printf("Used:\t%d KB\n", (int)pmc.PagefileUsage / 1024);
67          printf("WorkSet:%d KB\n", (int)pmc.WorkingSetSize / 1024);
68          printf("\n");
69
70          CloseHandle(hProcess);
71          bRet = Process32Next(hProcessSnap, &stcProcessInfo);
72      }
73      CloseHandle(hProcessSnap);
74      return ;
75  }
```

首先调用 CreateToolhelp32Snapshot()获取所有进程的快照，然后调用 Process32First()获取首个进程的信息，并调用 Process32Next()遍历所有

进程。对于遍历到的每个进程，通过 OpenProcess()根据其 pid 获取句柄，并通过通过 GetProcessMemoryInfo()获取该进程的内存信息。

- 查询单个进程的信息

```
// show process info by PID
void showSingleProcess() {
    HANDLE hProcessSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    HANDLE hProcess;

    if( hProcessSnap==INVALID_HANDLE_VALUE ) {
        printf("CreateToolhelp32Snapshot fialed\n");
        return ;
    }

    // get info for each process
    PROCESSENTRY32 stcProcessInfo;
    stcProcessInfo.dwSize = sizeof(stcProcessInfo);

    BOOL bRet = Process32First(hProcessSnap, &stcProcessInfo);
    PROCESS_MEMORY_COUNTERS pmc;
    pmc.cb = sizeof(pmc);

    int pid;
    printf("Input: ");
    scanf("%d", &pid);

    while (bRet) {
        if (pid == stcProcessInfo.th32ProcessID) {
            printf("PName: %s\n", stcProcessInfo.szExeFile);
            printf("PID: %d\n", stcProcessInfo.th32ProcessID);
            printf("ThreadNum: %d\n", stcProcessInfo.cntThreads);

            // get the memory used status about this process
            HANDLE hProcess =  OpenProcess(PROCESS_ALL_ACCESS, TRUE, stcProcessInfo.th32Pr

            GetProcessMemoryInfo(hProcess, &pmc, sizeof(pmc));
            printf("Used:\t%d KB\n", (int)pmc.PagefileUsage / 1024);
            printf("WorkSet:%d KB\n", (int)pmc.WorkingSetSize / 1024);
            printf("\n");

            WalkVM(hProcess);

            CloseHandle(hProcess);
            break;
        }
        // get the next handle about the process
        bRet = Process32Next(hProcessSnap, &stcProcessInfo);
    }
    CloseHandle(hProcessSnap);
}
```

查询单个进程信息与查询所有进程信息基本一致，只增加了两个新功能：1.判断进程 pid 是否为用户查询的 pid，若是则输出相应信息；2.显示被查询进程的虚拟地址空间布局。

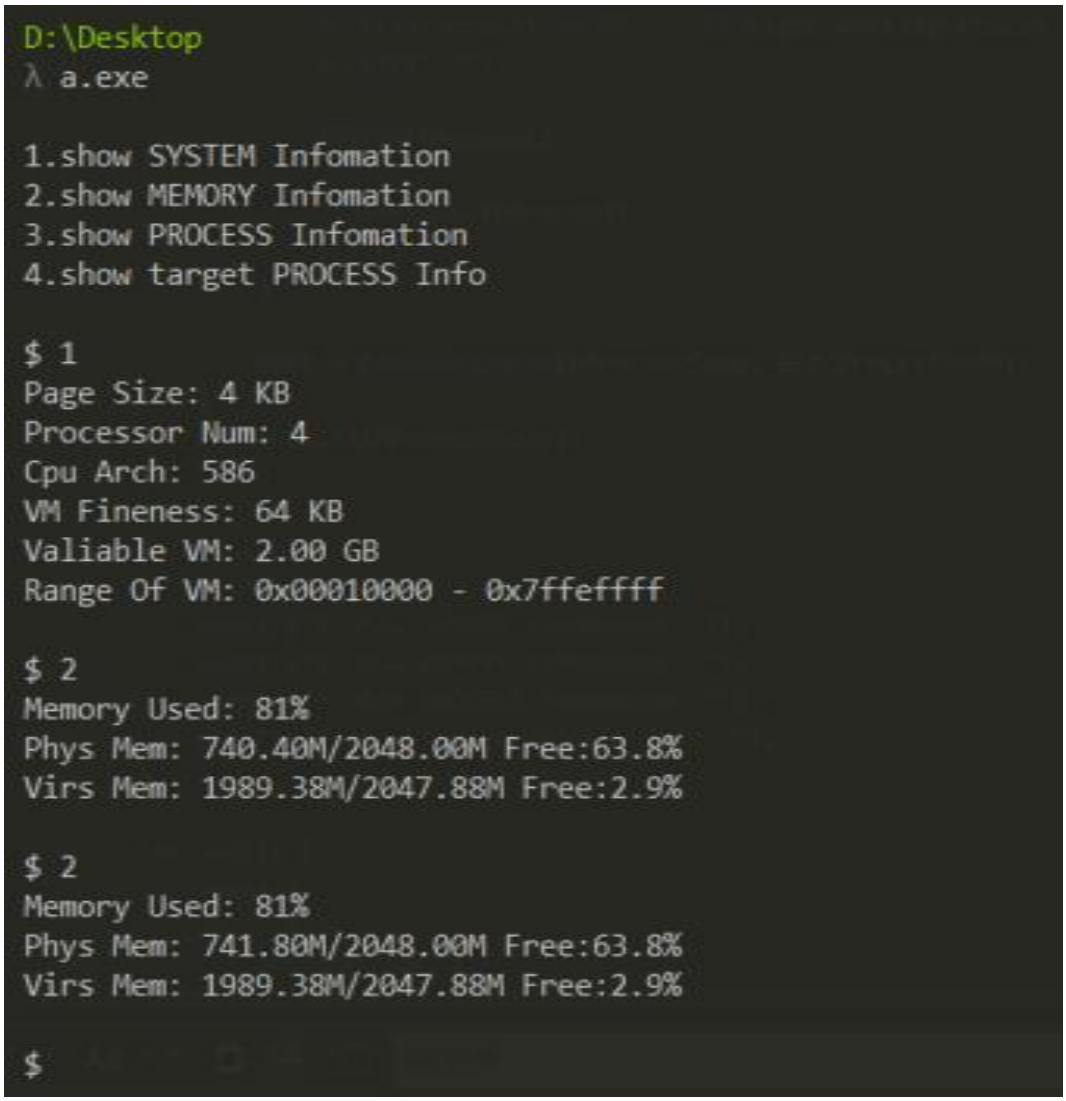
查询进程虚拟地址空间布局的函数 WalkVM()实现方法如下：

```
// showVM address
void WalkVM(HANDLE hProcess) {
    SYSTEM_INFO si;
    ZeroMemory(&si, sizeof(si));
    GetSystemInfo(&si);
    MEMORY_BASIC_INFORMATION mbi;
    ZeroMemory(&mbi, sizeof(mbi));

    LPCVOID pBlock = (LPVOID)si.lpMinimumApplicationAddress;

    while (pBlock < si.lpMaximumApplicationAddress) {
        if (VirtualQueryEx(hProcess,pBlock,&mbi,sizeof(mbi)) == sizeof(mbi)) {

            LPCVOID pEnd = (PBYTE)pBlock + mbi.RegionSize;
            TCHAR szSize[MAX_PATH];

            // show
            printf("0x%08X -- 0x%08X: ",pBlock, pEnd);

            // show the status about the block in the process
            switch (mbi.State) {
                case MEM_COMMIT:printf(" Committed"); break;
                case MEM_FREE:printf(" Free"); break;
                case MEM_RESERVE:printf(" Reserved"); break;
            }

            // show the type
            switch (mbi.Type) {
                case MEM_IMAGE:printf(", Image"); break;
                case MEM_MAPPED:printf(", Mapped"); break;
                case MEM_PRIVATE:printf(", Private"); break;
            }

            printf("\n");
            // get the next block
            pBlock = pEnd;
        }
    }
}

```

通过调用 GetSystemInfo()获取系统信息中的虚拟内存起始和终止地址，然后遍历整个地址空间，通过调用 VirtualQueryEx()，获取每块内存区域的信息，输出其起止地址、状态、类型等信息。

四、实验结果

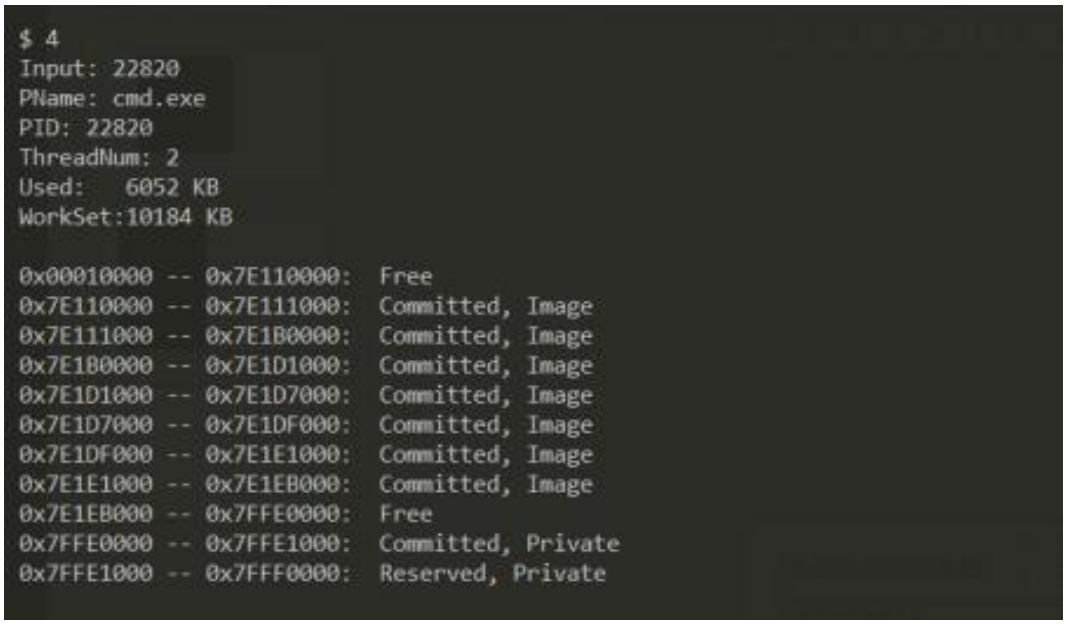
查询系统地址空间的布局、物理内存的使用情况的截图如下：



查询所有进程信息的截图如下(部分)：

```
Name:   svchost.exe
PID:    1376
Threads:10
PPID:   764
Used:   8084 KB
WorkSet:22500 KB

Name:   PresentationFontCache.exe
PID:    8160
Threads:4
PPID:   764
Used:   8084 KB
WorkSet:22500 KB

Name:   taskhostw.exe
PID:    1504
Threads:15
PPID:   520
Used:   8124 KB
WorkSet:14400 KB

Name:   RuntimeBroker.exe
PID:    784
Threads:16
PPID:   868
Used:   18736 KB
WorkSet:41104 KB

Name:   explorer.exe
PID:    8252
Threads:94
PPID:   4344
Used:   94260 KB
WorkSet:130756 KB

Name:   igfxEM.exe
PID:    8444
Threads:4
PPID:   8332
Used:   4928 KB
WorkSet:6564 KB

Name:   igfxHK.exe
PID:    8472
Threads:2
PPID:   8332
Used:   4064 KB
WorkSet:5252 KB
```

查询单个进程的信息结果，以 pid=22820 的进程为例，截图如下：



五、实验总结

通过实验熟悉并理解了有关获取系统信息、进程内存信息的 WIN32API 的使用方法，同时，对 windows 操作系统的地址空间有了更加深入的理解。