

实验名称	内存监视		
学号	1120141831	姓名	朴泉宇
<h3>一、实验目的</h3> <p>在 Windows 以及 Linux 平台下对系统信息、执行信息、内存信息以及对单个进程的监视。</p> <h3>二、实验内容</h3> <ol style="list-style-type: none">1. 在 Windows 下编写监视器代码2. 在 Windows 下编译运行，并监视各项信息3. 在 Linux 下直接调用系统命令，监视各项信息 <h3>三、实验环境及配置方法</h3> <ol style="list-style-type: none">1. 利用 Visual Studio 编写 Windows 下的代码，并编译运行2. 在 Ubuntu 中用 Terminal 调用系统命令 <h3>四、实验方法和实验步骤（程序设计与实现）</h3> <ol style="list-style-type: none">1. 编写 Windows 下的监视器代码<ol style="list-style-type: none">(1) 思路<p>此程序整体分成四个模块：</p><pre>1. void ShowSysInfo(); 2. void ShowPerformanceInfo(); 3. void ShowPhysicalMemoryInfo(); 4. void ShowTHISProcInfo();</pre><p>以及其中 <code>void ShowTHISProcInfo();</code> 模块调用的以下两个子模块：</p><pre>1. void ShowVirtualMemInfo(HANDLE hProcess); 2. void ShowWorkingSetInfo(int PID);</pre><p>以及为实时覆盖输出的有关控制台光标位置的函数：</p><pre>1. void GetConsoleCursorXY(int &x, int &y); 2. void SetConsoleCursorTo(int x, int y);</pre>			

操作系统课程设计实验报告

1) `void ShowSysInfo();`模块:

创建 `SYSTEM_INFO` 结构体变量, 并调用 `GetSystemInfo();` 来实时获取操作系统的各项信息。

2) `void ShowPerformanceInfo();`模块:

创建 `PERFORMANCE_INFORMATION` 结构体变量, 并调用 `GetPerformanceInfo();` 来实时获取操作系统的执行信息。

3) `void ShowPhysicalMemoryInfo();`模块:

创建 `MEMORYSTATUSEX` 结构体变量, 并调用 `GlobalMemoryStatusEx();` 来实时获取当前物理内存的使用情况。

4) `void ShowTHISProcInfo();`模块:

通过 Windows 自带的任务管理器获取某进程的 PID, 通过 PID 获得此进程的 HANDLE。

`void ShowVirtualMemInfo();` 子模块创建 `SYSTEM_INFO` 和 `MEMORY_BASIC_INFORMATION` 结构体变量, 并调用 `GetSystemInfo();` 和 `VirtualQueryEx();` 来实时获取进程的虚拟内存使用情况。

`void ShowWorkingSetInfo();` 子模块创建 `PROCESSENTRY32` 和 `PROCESS_MEMORY_COUNTERS` 结构体变量, 并调用 `Process32First();` 和 `GetProcessMemoryInfo();` 来实时获取此进程的工作集情况。

5) `void GetConsoleCursorXY();`函数:

获取当前控制台的光标位置。与 `kbhit();` 配合, 完成时事刷新信息的功能。

6) `void SetConsoleCursorTo();`函数:

将控制台的光标设定在指定位置。与 `kbhit();` 配合, 完成时事刷新信息的功能。

(2) 函数解释

1) `void ShowSysInfo();`模块:

```
1. /*Show System Info*/
2. void ShowSysInfo()
3. {
4.     SYSTEM_INFO SystemInfo;
```

操作系统课程设计实验报告

```
5.     int X = 0, Y = 0, flag = 0; //Cursor Position
6.
7.     ZeroMemory(&SystemInfo, sizeof(SystemInfo));
8.
9.     while (kbhit() == 0)
10.    {
11.        if (flag == 0)
12.        {
13.            GetConsoleCursorXY(X, Y);
14.            flag = 1;
15.        }
16.        else
17.        {
18.            SetConsoleCursorTo(X, Y);
19.        }
20.
21.        GetSystemInfo(&SystemInfo);
22.
23.        printf("Hardware information:\n");
24.        printf("OEM ID: %u\n", SystemInfo.dwOemId); //1. OEM ID
25.        printf("Processor Architecture: "); //2. Processor Architecture
26.        switch (SystemInfo.wProcessorArchitecture)
27.        {
28.            case 0:printf("x86\n");break;
29.            case 5:printf("ARM\n");break;
30.            case 6:printf("Intel Itanium-based\n");break;
31.            case 9:printf("x64 (AMD or Intel)\n");break;
32.            case 0xffff:printf("Unknown architecture.\n");break;
33.        }
34.        printf("Page Size(Bytes): %u\n", SystemInfo.dwPageSize); //3. Page Size
35.
36.        printf("Minimum Application Address: 0x%lx\n", SystemInfo.lpMinimumApplica
            tionAddress); //4. Minimum Application Address
37.        printf("Maximum Application Address: 0x%lx\n", SystemInfo.lpMaximumApplica
            tionAddress); //5. Maximum Application Address
38.        printf("Size of Application's usable virtual memory(Bytes): %u\n", ((DWORD
            )SystemInfo.lpMaximumApplicationAddress - (DWORD)SystemInfo.lpMinimumApplicationAd
            dress)); //Extend. Size of Application's usable virtual memory
39.        printf("Active Processor Mask: %u\n", SystemInfo.dwActiveProcessorMask);
            //6. Active Processor Mask
40.        printf("Number of Processors: %u\n", SystemInfo.dwNumberOfProcessors); //
            7. Number of Processors
41.        printf("Processor Type: %u\n", SystemInfo.dwProcessorType); //8. Processor
            Type
```

操作系统课程设计实验报告

```
41.     printf("Allocation Granularity: %u\n", SystemInfo.dwAllocationGranularity)
; //9. Allocation Granularity
42.     printf("Processor Level: %u\n", SystemInfo.wProcessorLevel);    //10. Proc
    essor Level
43.     printf("Processor Revision: %u\n", SystemInfo.wProcessorRevision); //11.
    "Processor Revision
44.
45.     printf("\n*****Press <ESC> to stop.*****\n");
46.
47.     Sleep(1000);    //Refresh per second.
48. }
49. }
```

2) `void ShowPerformanceInfo()`;模块:

```
1. /*Show Performance Info*/
2. void ShowPerformanceInfo()
3. {
4.     PERFORMANCE_INFORMATION PerformanceInfo;
5.     int X = 0, Y = 0, flag = 0; //Cursor Position
6.
7.     ZeroMemory(&PerformanceInfo, sizeof(PerformanceInfo));
8.     PerformanceInfo.cb = sizeof(PerformanceInfo);
9.
10.    while (kbhit() == 0)
11.    {
12.        if (flag == 0)
13.        {
14.            GetConsoleCursorXY(X, Y);
15.            flag = 1;
16.        }
17.        else
18.        {
19.            SetConsoleCursorTo(X, Y);
20.        }
21.
22.        GetPerformanceInfo(&PerformanceInfo, PerformanceInfo.cb);
23.
24.        printf("CommitTotal(pages): %u\n", PerformanceInfo.CommitTotal);    //1. T
        he number of pages currently committed by the system.
25.        printf("CommitLimit(pages): %u\n", PerformanceInfo.CommitLimit);    //2. T
        he current maximum number of pages that can be committed by the system without ext
        ending the paging file(s).
26.        printf("CommitPeak(pages): %u\n", PerformanceInfo.CommitPeak);    //3. The m
```

操作系统课程设计实验报告

```
aximum number of pages that were simultaneously in the committed state since the last system reboot.
27.     printf("PhysicalTotal(pages): %u\n", PerformanceInfo.PhysicalTotal);    //
4. The amount of actual physical memory, in pages.
28.     printf("PhysicalAvailable(pages): %u\n", PerformanceInfo.PhysicalAvailable); //5. The amount of physical memory currently available, in pages.
29.     printf("SystemCache(pages): %u\n", PerformanceInfo.SystemCache);    //6. The amount of system cache memory, in pages.
30.     printf("KernelTotal(pages): %u\n", PerformanceInfo.KernelTotal);    //7. The sum of the memory currently in the paged and nonpaged kernel pools, in pages.
31.     printf("KernelPaged(pages): %u\n", PerformanceInfo.KernelPaged);    //8. The memory currently in the paged kernel pool, in pages.
32.     printf("KernelNonpaged(pages): %u\n", PerformanceInfo.KernelNonpaged); //9. The memory currently in the nonpaged kernel pool, in pages.
33.     printf("PageSize(Bytes): %u\n", PerformanceInfo.PageSize); //10. The size of a page, in bytes.
34.     printf("HandleCount: %u\n", PerformanceInfo.HandleCount); //11. The current number of open handles.
35.     printf("ProcessCount: %u\n", PerformanceInfo.ProcessCount); //12. The current number of processes.
36.     printf("ThreadCount: %u\n", PerformanceInfo.ThreadCount); //13. The current number of threads.
37.
38.     printf("\n*****Press <ESC> to stop.*****\n");
39.
40.     Sleep(1000);    //Refresh per second.
41. }
42. }
```

3) `void ShowPhysicalMemoryInfo()`;模块:

```
1. /*Show Physical Memory Info*/
2. void ShowPhysicalMemoryInfo()
3. {
4.     MEMORYSTATUSEX MemoryStatusEX;
5.     int X = 0, Y = 0, flag = 0; //Cursor Position
6.
7.     ZeroMemory(&MemoryStatusEX, sizeof(MemoryStatusEX));
8.     MemoryStatusEX.dwLength = sizeof(MemoryStatusEX);
9.
10.    while (kbhit() == 0)
11.    {
12.        if (flag == 0)
13.        {
```

```

14.         GetConsoleCursorXY(X, Y);
15.         flag = 1;
16.     }
17.     else
18.     {
19.         SetConsoleCursorTo(X, Y);
20.     }
21.
22.     GlobalMemoryStatusEx(&MemoryStatusEX);
23.
24.     printf("There is %ld percent of memory in use.\n", MemoryStatusEX.dwMemory
        Load);
25.     printf("There are %I64d total KB of physical memory.\n", MemoryStatusEX.ul
        lTotalPhys / 1024);
26.     printf("There are %I64d free KB of physical memory.\n", MemoryStatusEX.ul
        lAvailPhys / 1024);
27.     printf("There are %I64d total KB of paging file.\n", MemoryStatusEX.ul
        lTotalPageFile / 1024);
28.     printf("There are %I64d free KB of paging file.\n", MemoryStatusEX.ul
        lAvailPageFile / 1024);
29.     printf("There are %I64d total KB of virtual memory.\n", MemoryStatusEX.ul
        lTotalVirtual / 1024);
30.     printf("There are %I64d free KB of virtual memory.\n", MemoryStatusEX.ul
        lAvailVirtual / 1024);
31.
32.     // Show the amount of extended memory available.
33.     printf("There are %I64d free KB of extended memory.\n", MemoryStatusEX.ul
        lAvailExtendedVirtual / 1024);
34.
35.     printf("\n*****Press <ESC> to stop.*****\n");
36.
37.     Sleep(1000);    //Refresh per second.
38. }
39. }

```

4) `void` ShowTHISProcInfo();模块:

```

1. /*Show THIS Process Info, includes Virtual Memory Info and Working Set Info*/
2. void ShowTHISProcInfo()
3. {
4.     int THISPID = MonitorPID;
5.     HANDLE hTHISProcess;
6.     int X = 0, Y = 0, flag = 0; //Cursor Position
7.

```

操作系统课程设计实验报告

```
8.     //THISPID = GetCurrentProcessId();
9.     hTHISProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, THISPID);
10.
11.     while (kbhit() == 0)
12.     {
13.         if (hTHISProcess != NULL)
14.         {
15.             if (flag == 0)
16.             {
17.                 GetConsoleCursorXY(X, Y);
18.                 flag = 1;
19.             }
20.             else
21.             {
22.                 SetConsoleCursorTo(X, Y);
23.             }
24.
25.             ShowVirtualMemInfo(hTHISProcess);
26.             ShowWorkingSetInfo(THISPID);
27.
28.             printf("\n*****Press <ESC> to stop.*****\n");
29.             printf("*****Close the monitor first, then close The Process.*****\n")
30.             ;
31.             Sleep(1000);    //Refresh per second.
32.         }
33.         else
34.         {
35.             printf("The Process was not exist. The monitor will close.\n");
36.
37.             return;
38.         }
39.     }
40. }
```

5) `void ShowVirtualMemInfo()`;子模块:

```
1.  /*Show THE PROCESS's Virtual Memory Info*/
2.  void ShowVirtualMemInfo(HANDLE hProcess)
3.  {
4.      SYSTEM_INFO SI;
5.      MEMORY_BASIC_INFORMATION MBI;
6.
7.      ZeroMemory(&SI, sizeof(SI));
```

操作系统课程设计实验报告

```
8.     ZeroMemory(&MBI, sizeof(MBI));
9.
10.    GetSystemInfo(&SI);
11.    LPCVOID pBlock = (LPVOID)SI.lpMinimumApplicationAddress;
12.
13.    while (pBlock < SI.lpMaximumApplicationAddress)
14.    {
15.        if (VirtualQueryEx(hProcess, pBlock, &MBI, sizeof(MBI)) == sizeof(MBI))
16.        {
17.            LPCVOID pEnd = (PBYTE)pBlock + MBI.RegionSize;
18.            printf("Block starts at: 0x%lx, ", pBlock); //1. Show Block Starts Addr.
19.            printf("Block Length(Bytes): %u, ", MBI.RegionSize); //2. Show Block's length.
20.            switch (MBI.State) //3. Show Block's State.
21.            {
22.                case MEM_COMMIT:
23.                    printf("Committed, ");break;
24.                case MEM_FREE:
25.                    printf("Free.\n");break;
26.                case MEM_RESERVE:
27.                    printf("Reserved, ");break;
28.            }
29.            switch (MBI.Type) //4. Show Block's Type.
30.            {
31.                case MEM_IMAGE:
32.                    printf("Image.\n");break;
33.                case MEM_MAPPED:
34.                    printf("Mapped.\n");break;
35.                case MEM_PRIVATE:
36.                    printf("Private.\n");break;
37.            }
38.
39.            pBlock = pEnd;
40.        }
41.        else
42.        {
43.            break;
44.        }
45.    }
46. }
```

6) void ShowWorkingSetInfo();子模块:


```

1.  /*Show THE PROCESS's Working Set Info*/
2.  void ShowWorkingSetInfo(int PID)
3.  {
4.      PROCESSENTRY32 PE32;
5.      PROCESS_MEMORY_COUNTERS ProcMemCounter;
6.      HANDLE hProcessSnap;
7.
8.      ZeroMemory(&PE32, sizeof(PE32));
9.      ZeroMemory(&ProcMemCounter, sizeof(ProcMemCounter));
10.     PE32.dwSize = sizeof(PE32);
11.     ProcMemCounter.cb = sizeof(ProcMemCounter);
12.
13.     hProcessSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
14.
15.     BOOL bNext = Process32First(hProcessSnap, &PE32);
16.     printf("Process Working Set Information:\n");
17.     while (bNext)
18.     {
19.         if (PID == PE32.th32ProcessID)
20.         {
21.             HANDLE hProcess;
22.             hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE, PID);
23.             GetProcessMemoryInfo(hProcess, &ProcMemCounter, sizeof(ProcMemCounter)
24.             );
25.             printf("Committed(Bytes): %u\n", ProcMemCounter.PagefileUsage);
26.             printf("Working Set Size(Byte): %u\n", ProcMemCounter.WorkingSetSize);
27.
28.             printf("Peak Working Set Size(Byte): %u\n", ProcMemCounter.PeakWorking
29.             SetSize);
30.
31.             break;
32.         }
33.     }
34.     bNext = Process32Next(hProcessSnap, &PE32);
35. }

```

7) void GetConsoleCursorXY();函数:

```

1.  /*Get console cursor's position*/
2.  void GetConsoleCursorXY(int &x, int &y)
3.  {
4.      HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

```

```

5.     CONSOLE_SCREEN_BUFFER_INFO CSBI;
6.
7.     GetConsoleScreenBufferInfo(hConsole, &CSBI);
8.
9.     x = CSBI.dwCursorPosition.X;
10.    y = CSBI.dwCursorPosition.Y;
11. }

```

8) `void SetConsoleCursorTo();`函数:

```

1.  /*Set console cursor to (x, y)*/
2.  void SetConsoleCursorTo(int x, int y)
3.  {
4.      HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
5.      COORD SetCursorPosition = { x, y };
6.
7.      SetConsoleCursorPosition(hConsole, SetCursorPosition);
8.  }

```

(3) `main` 函数

`main` 函数起到引导用户选择并使用 `Monitor` 的作用。

```

1.  int main()
2.  {
3.      Restart:
4.      printf("Please Input the PID to monitor a process.\n");
5.      printf("If don't want to monitor any process, please input -1.\n");
6.      scanf("%d", &MonitorPID);
7.
8.      if (MonitorPID == -1)
9.      {
10.         int SelectNum;
11.
12.         printf("1. Show System Information.\n");
13.         printf("2. Show Performance Infomation.\n");
14.         printf("3. Show Physical Memory Information.\n");
15.         printf("Please input the number to select.\n");
16.
17.         scanf("%d", &SelectNum);
18.         switch (SelectNum)
19.         {
20.             case 1: ShowSysInfo(); break;
21.             case 2: ShowPerformanceInfo(); break;

```

操作系统课程设计实验报告

```
22.     case 3:ShowPhysicalMemoryInfo();break;
23.     default:printf("Input Error!\n");break;
24.     }
25.
26.     printf("\nPlease input the command:\n");
27.     printf("[0] Restart the monitor.\n");
28.     printf("Other key to close the monitor.\n");
29.     int command = -1;
30.     scanf("%d", &command);
31.     if (command == 0)
32.     {
33.         goto Restart;
34.     }
35.     else
36.     {
37.         printf("Thank you for using.\n");
38.     }
39. }
40. else
41. {
42.     int SelectNum;
43.
44.     printf("1. Show System Information.\n");
45.     printf("2. Show Performance Infomation.\n");
46.     printf("3. Show Physical Memory Information.\n");
47.     printf("4. Show This Process Information, includes Virtual Memory Informat
ion and Working Set Information.\n");
48.     printf("Please input the number to select.\n");
49.
50.     scanf("%d", &SelectNum);
51.     switch (SelectNum)
52.     {
53.         case 1:ShowSysInfo();break;
54.         case 2:ShowPerformanceInfo();break;
55.         case 3:ShowPhysicalMemoryInfo();break;
56.         case 4:ShowTHISProcInfo();break;
57.         default:printf("Input Error!\n");break;
58.     }
59.
60.     printf("\nPlease input the command:\n");
61.     printf("[0] Restart the monitor.\n");
62.     printf("Other key to close the monitor.\n");
63.     int command = -1;
64.     scanf("%d", &command);
```

```
65.     if (command == 0)
66.     {
67.         goto Restart;
68.     }
69.     else
70.     {
71.         printf("Thank you for using.\n");
72.     }
73. }
74. }
```

2. Linux 下直接调用系统命令

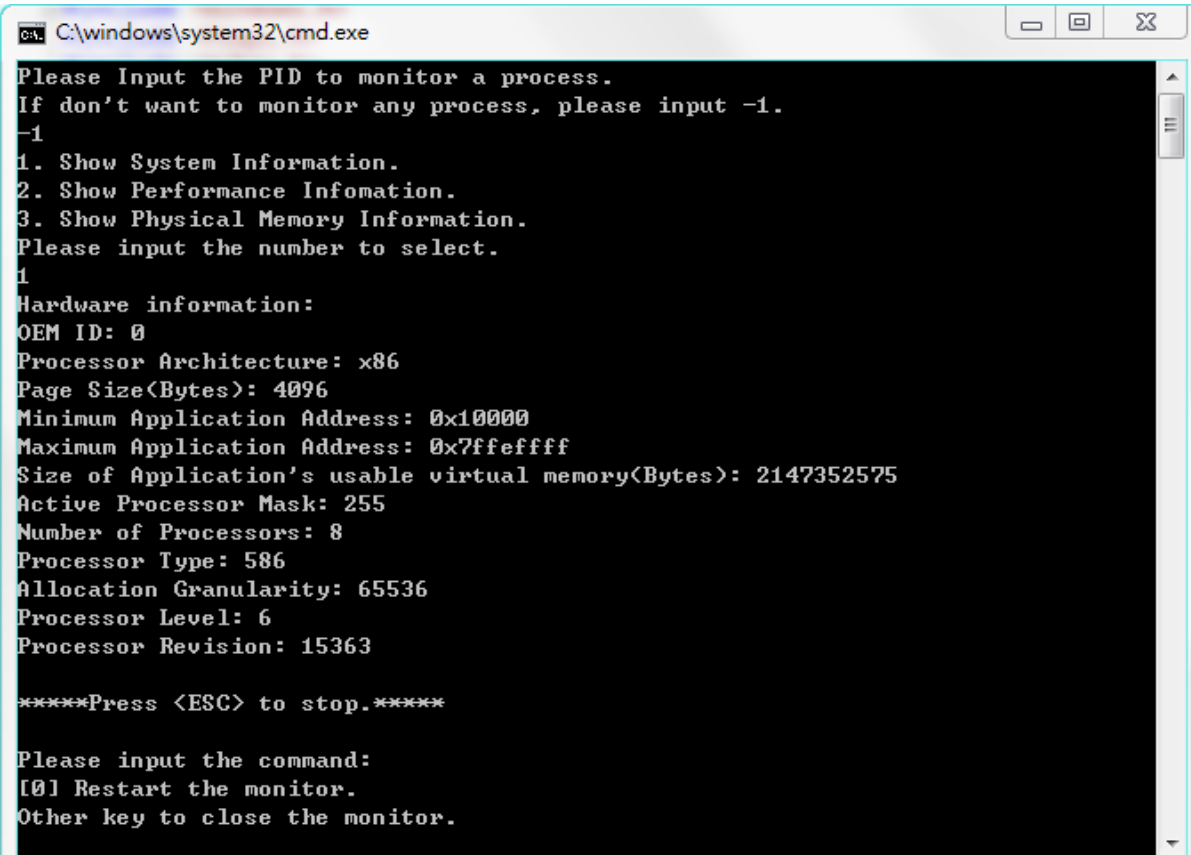
(1) 思路

- 1) 了解 Linux 的 top 命令以及打开 top 之后的 P（按 CPU 使用率）、T（按进程时间累计）、M（按内存占用率）等排序命令。
- 2) 使用 ps -A 查看所有进程，并找到指定进程的 PID。
- 3) 使用 top -p PID 查看指定进程的情况。
- 4) 使用 pmap -d PID 查看指定进程的内存使用情况。
- 5) 额外了解 htop 命令，并对比、使用 htop 完成如 Windows 下任务管理器的操作。

(2) 实验过程略

五、实验结果和分析

1. Windows 下的实验结果



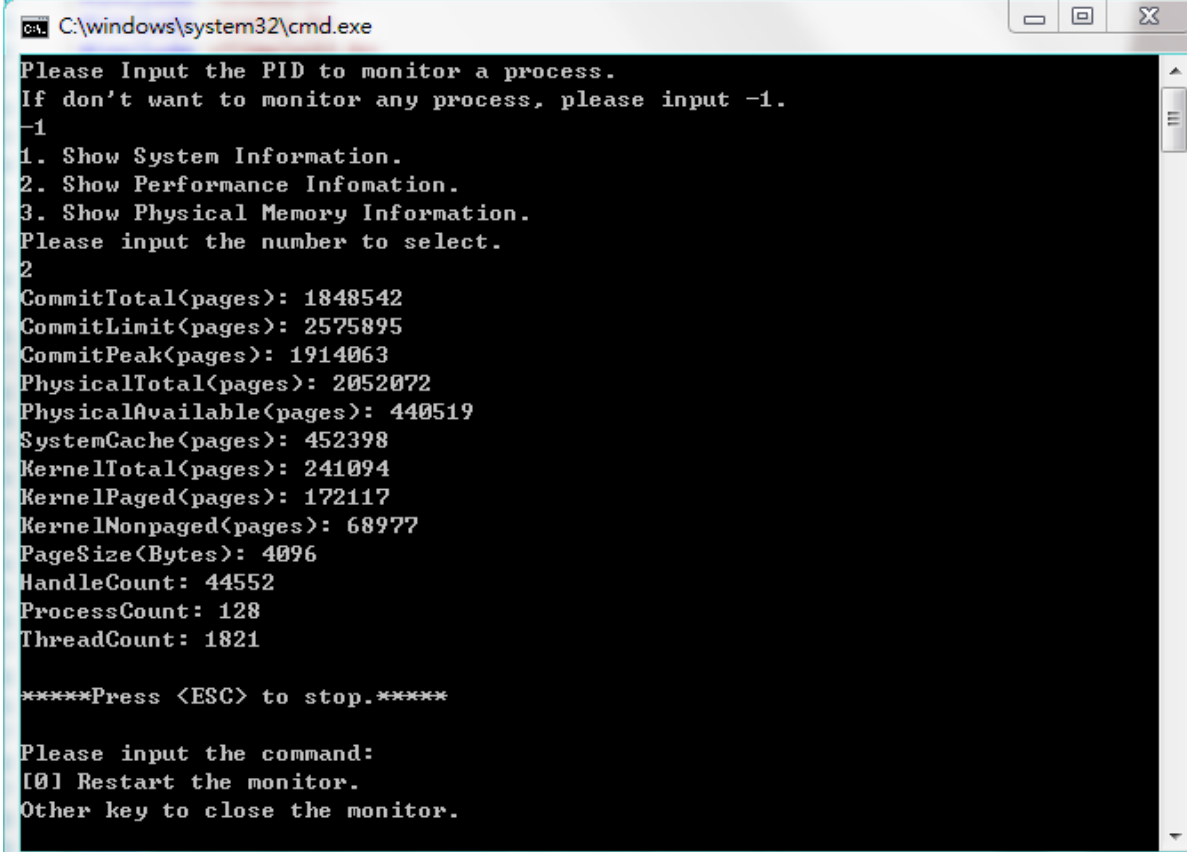
```
C:\windows\system32\cmd.exe

Please Input the PID to monitor a process.
If don't want to monitor any process, please input -1.
-1
1. Show System Information.
2. Show Performance Infomation.
3. Show Physical Memory Information.
Please input the number to select.
1
Hardware information:
OEM ID: 0
Processor Architecture: x86
Page Size(Bytes): 4096
Minimum Application Address: 0x10000
Maximum Application Address: 0x7ffeffff
Size of Application's usable virtual memory(Bytes): 2147352575
Active Processor Mask: 255
Number of Processors: 8
Processor Type: 586
Allocation Granularity: 65536
Processor Level: 6
Processor Revision: 15363

*****Press <ESC> to stop.*****

Please input the command:
[0] Restart the monitor.
Other key to close the monitor.
```

(图 1) `void ShowSysInfo();`模块



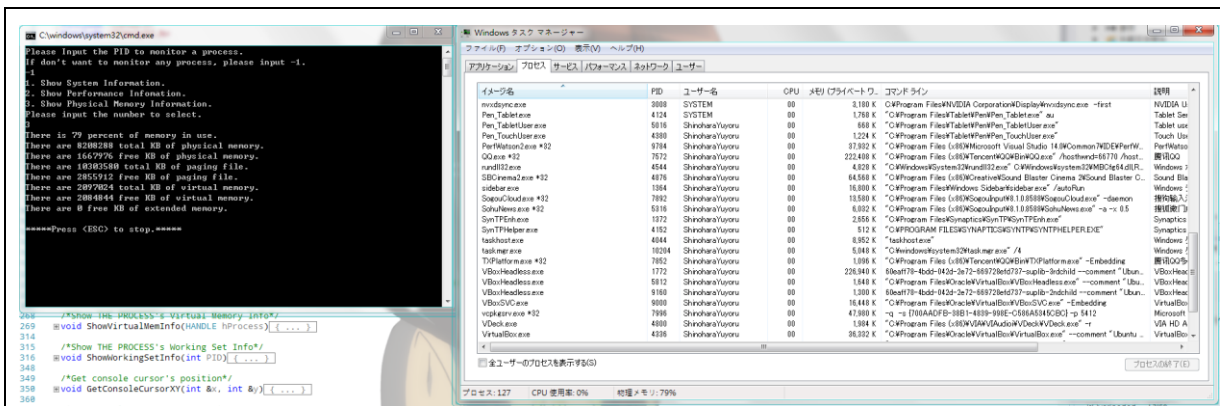
```
C:\windows\system32\cmd.exe
Please Input the PID to monitor a process.
If don't want to monitor any process, please input -1.
-1
1. Show System Information.
2. Show Performance Infomation.
3. Show Physical Memory Information.
Please input the number to select.
2
CommitTotal(pages): 1848542
CommitLimit(pages): 2575895
CommitPeak(pages): 1914063
PhysicalTotal(pages): 2052072
PhysicalAvailable(pages): 440519
SystemCache(pages): 452398
KernelTotal(pages): 241094
KernelPaged(pages): 172117
KernelNonpaged(pages): 68977
PageSize(Bytes): 4096
HandleCount: 44552
ProcessCount: 128
ThreadCount: 1821

*****Press <ESC> to stop.*****

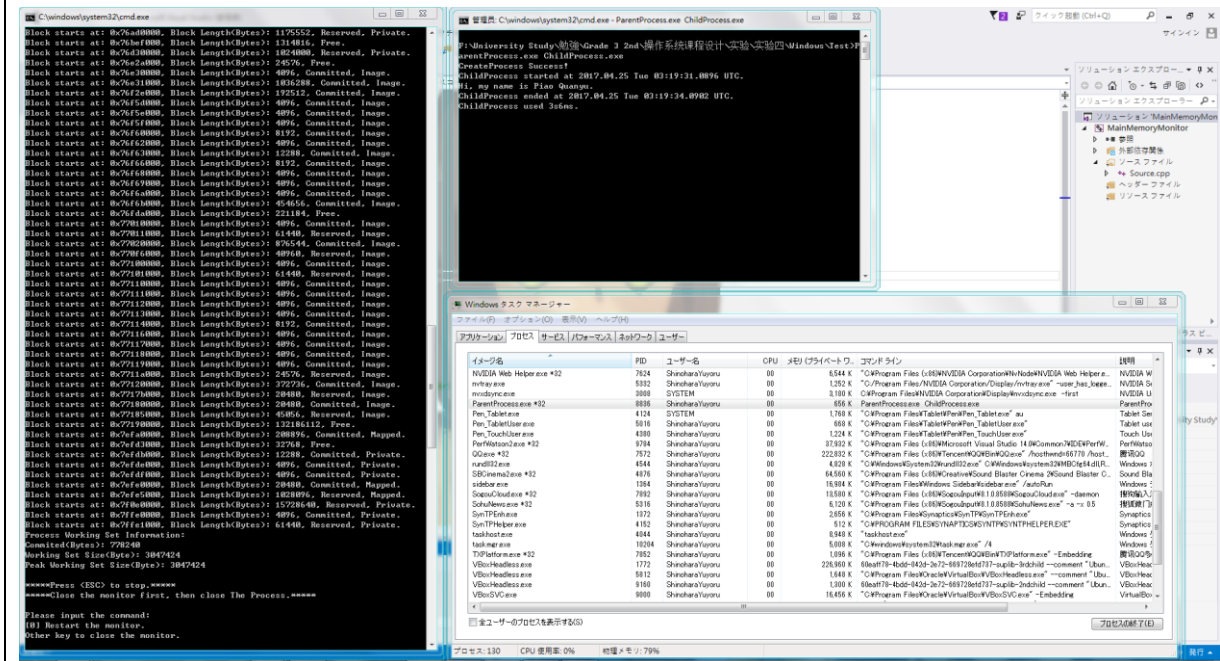
Please input the command:
[0] Restart the monitor.
Other key to close the monitor.
```

(图 2) `void ShowPerformanceInfo();`模块

操作系统课程设计实验报告



(图 3) `void ShowPhysicalMemoryInfo();` 模块



(图 4) `void ShowTHISProcInfo();` 模块

2. Linux 下的实验结果

```
root@ShYy-VirtualBox: ~  
top - 11:23:09 up 1:13, 1 user, load average: 0.22, 0.07, 0.06  
Tasks: 199 total, 1 running, 198 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 4.2 us, 1.4 sy, 0.0 ni, 88.6 id, 5.8 wa, 0.0 hi, 0.0 si, 0.0 st  
KiB Mem : 2048128 total, 483608 free, 612028 used, 952492 buff/cache  
KiB Swap: 0 total, 0 free, 0 used. 1370872 avail Mem  
  
  PID USER      PR  NI    VIRT    RES    SHR  S  %CPU  %MEM    TIME+  COMMAND  
 2582 root        20   0   974652   58548  43116 S   8.0   2.9   0:03.85 nautilus  
 1918 root        20   0  1244620   87420  52532 S   6.6   4.3   0:27.95 compiz  
   921 root        20   0   457432  103424  41916 S   6.3   5.0   0:25.10 Xorg  
 1915 root        20   0   636388   41396  31464 S   1.7   2.0   0:00.60 hud-service  
 3815 root        20   0   663240   36600  28128 S   1.3   1.8   0:00.74 gnome-terminal-  
 1893 root        20   0   525604   29796  22088 S   1.0   1.5   0:00.86 bamfdaemon  
 1766 root        20   0    43736    4268   2936 S   0.7   0.2   0:00.45 dbus-daemon  
    7 root        20   0         0         0        0 S   0.3   0.0   0:00.37 rcu_sched  
   23 root        20   0         0         0        0 S   0.3   0.0   0:00.06 ksoftirqd/3  
   603 message+  20   0    44192    5268   3656 S   0.3   0.3   0:00.55 dbus-daemon  
  1645 root        20   0   139916    2068   1636 S   0.3   0.1   0:04.66 VBoxClient  
  1814 root        20   0   365396    8584   7224 S   0.3   0.4   0:00.85 ibus-daemon  
  1852 root        20   0   479432   29728  24708 S   0.3   1.5   0:00.28 ibus-ui-gtk3  
  2624 root        20   0  1507904  120676  27064 S   0.3   5.9   0:02.97 gnome-software  
 3833 root        20   0    49028    4108   3548 R   0.3   0.2   0:00.19 top  
    1 root        20   0   119664    5796   3968 S   0.0   0.3   0:01.43 systemd  
    2 root        20   0         0         0        0 S   0.0   0.0   0:00.00 kthreadd  
    3 root        20   0         0         0        0 S   0.0   0.0   0:00.01 ksoftirqd/0  
    5 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 kworker/0:0H  
    8 root        20   0         0         0        0 S   0.0   0.0   0:00.00 rcu_bh  
    9 root        rt    0         0         0        0 S   0.0   0.0   0:00.00 migration/0  
   10 root        rt    0         0         0        0 S   0.0   0.0   0:00.01 watchdog/0  
   11 root        rt    0         0         0        0 S   0.0   0.0   0:00.01 watchdog/1  
   12 root        rt    0         0         0        0 S   0.0   0.0   0:00.00 migration/1  
   13 root        20   0         0         0        0 S   0.0   0.0   0:00.02 ksoftirqd/1  
   15 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 kworker/1:0H  
   16 root        rt    0         0         0        0 S   0.0   0.0   0:00.01 watchdog/2  
   17 root        rt    0         0         0        0 S   0.0   0.0   0:00.00 migration/2  
   18 root        20   0         0         0        0 S   0.0   0.0   0:00.01 ksoftirqd/2  
   19 root        20   0         0         0        0 S   0.0   0.0   0:00.00 kworker/2:0  
   20 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 kworker/2:0H  
   21 root        rt    0         0         0        0 S   0.0   0.0   0:00.01 watchdog/3  
   22 root        rt    0         0         0        0 S   0.0   0.0   0:00.00 migration/3  
   25 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 kworker/3:0H  
   26 root        20   0         0         0        0 S   0.0   0.0   0:00.00 kdevtmpfs  
   27 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 netns  
   28 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 perf  
   29 root        20   0         0         0        0 S   0.0   0.0   0:00.00 khungtaskd  
   30 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 writeback  
   31 root        25   5         0         0        0 S   0.0   0.0   0:00.00 ksmd  
   32 root        39  19         0         0        0 S   0.0   0.0   0:00.13 khugepaged  
   33 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 crypto  
   34 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 kintegrityd  
   35 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 bioset  
   36 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 kblockd  
   37 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 ata_sff  
   38 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 md  
   39 root        0 -20         0         0        0 S   0.0   0.0   0:00.00 devfreq_wq
```

(图 1) top 命令，P（按 CPU 使用率）排序


```
root@ShYy-VirtualBox: ~  
top - 11:23:33 up 1:14, 1 user, load average: 0.14, 0.07, 0.06  
Tasks: 199 total, 1 running, 198 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 25.0 us, 0.0 sy, 0.0 ni, 75.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
KiB Mem : 2048128 total, 483672 free, 611944 used, 952512 buff/cache  
KiB Swap: 0 total, 0 free, 0 used. 1370936 avail Mem  


| PID  | USER     | PR | NI | VIRT    | RES    | SHR   | S | %CPU  | %MEM | TIME+   | COMMAND         |
|------|----------|----|----|---------|--------|-------|---|-------|------|---------|-----------------|
| 1918 | root     | 20 | 0  | 1244620 | 87420  | 52532 | S | 0.0   | 4.3  | 0:28.04 | compiz          |
| 921  | root     | 20 | 0  | 457432  | 103424 | 41916 | S | 0.0   | 5.0  | 0:25.24 | Xorg            |
| 1645 | root     | 20 | 0  | 139916  | 2068   | 1636  | S | 0.0   | 0.1  | 0:04.68 | VBoxClient      |
| 2582 | root     | 20 | 0  | 974652  | 58552  | 43116 | S | 0.0   | 2.9  | 0:03.85 | nautilus        |
| 2624 | root     | 20 | 0  | 1507904 | 120676 | 27064 | S | 0.0   | 5.9  | 0:02.97 | gnome-software  |
| 1    | root     | 20 | 0  | 119664  | 5796   | 3968  | S | 0.0   | 0.3  | 0:01.43 | systemd         |
| 1814 | root     | 20 | 0  | 365396  | 8584   | 7224  | S | 200.0 | 0.4  | 0:00.86 | ibus-daemon     |
| 1893 | root     | 20 | 0  | 525604  | 29796  | 22088 | S | 0.0   | 1.5  | 0:00.86 | bamfdemon       |
| 3815 | root     | 20 | 0  | 663240  | 36600  | 28128 | S | 0.0   | 1.8  | 0:00.80 | gnome-terminal- |
| 2691 | root     | 20 | 0  | 635456  | 36488  | 10792 | S | 0.0   | 1.8  | 0:00.62 | fwupd           |
| 1915 | root     | 20 | 0  | 636388  | 41396  | 31464 | S | 0.0   | 2.0  | 0:00.60 | hud-service     |
| 603  | message+ | 20 | 0  | 44192   | 5268   | 3656  | S | 0.0   | 0.3  | 0:00.55 | dbus-daemon     |
| 975  | root     | 20 | 0  | 263816  | 3112   | 2656  | S | 0.0   | 0.2  | 0:00.53 | VBoxService     |
| 1766 | root     | 20 | 0  | 43736   | 4268   | 2936  | S | 0.0   | 0.2  | 0:00.45 | dbus-daemon     |
| 3605 | root     | 20 | 0  | 0       | 0      | 0     | S | 0.0   | 0.0  | 0:00.41 | kworker/3:1     |
| 7    | root     | 20 | 0  | 0       | 0      | 0     | S | 0.0   | 0.0  | 0:00.37 | rcu_sched       |
| 1892 | root     | 20 | 0  | 864252  | 31460  | 24512 | S | 0.0   | 1.5  | 0:00.36 | unity-settings- |
| 1852 | root     | 20 | 0  | 479432  | 29728  | 24708 | S | 0.0   | 1.5  | 0:00.28 | ibus-ui-gtk3    |
| 1882 | root     | 20 | 0  | 208820  | 8208   | 7460  | S | 0.0   | 0.4  | 0:00.25 | ibus-engine-sim |
| 1908 | root     | 20 | 0  | 564972  | 32816  | 25528 | S | 0.0   | 1.6  | 0:00.23 | unity-panel-ser |
| 2688 | root     | 20 | 0  | 869356  | 62292  | 22472 | S | 0.0   | 3.0  | 0:00.22 | evolution-calen |
| 3833 | root     | 20 | 0  | 49028   | 4108   | 3548  | R | 100.0 | 0.2  | 0:00.22 | top             |
| 263  | root     | 20 | 0  | 45344   | 4588   | 3084  | S | 0.0   | 0.2  | 0:00.21 | systemd-udev    |
| 2724 | root     | 20 | 0  | 829480  | 51364  | 15084 | S | 0.0   | 2.5  | 0:00.21 | evolution-calen |
| 2587 | root     | 20 | 0  | 590972  | 34336  | 27228 | S | 0.0   | 1.7  | 0:00.17 | nm-applet       |
| 3486 | root     | 20 | 0  | 449116  | 28868  | 22448 | S | 0.0   | 1.4  | 0:00.14 | notify-osd      |
| 32   | root     | 39 | 19 | 0       | 0      | 0     | S | 0.0   | 0.0  | 0:00.13 | khugepaged      |
| 582  | root     | 20 | 0  | 298368  | 8692   | 7780  | S | 0.0   | 0.4  | 0:00.13 | accounts-daemon |
| 1982 | root     | 20 | 0  | 651040  | 26636  | 21472 | S | 0.0   | 1.3  | 0:00.13 | indicator-keybo |
| 236  | root     | 20 | 0  | 29756   | 3620   | 3272  | S | 0.0   | 0.2  | 0:00.12 | systemd-journal |
| 680  | root     | 20 | 0  | 462524  | 16972  | 14512 | S | 0.0   | 0.8  | 0:00.12 | NetworkManager  |
| 1445 | colord   | 20 | 0  | 320644  | 13164  | 10356 | S | 0.0   | 0.6  | 0:00.12 | colord          |
| 1603 | root     | 20 | 0  | 53516   | 5192   | 4028  | S | 0.0   | 0.3  | 0:00.12 | upstart         |
| 2736 | root     | 20 | 0  | 1076004 | 52524  | 14132 | S | 0.0   | 2.6  | 0:00.12 | evolution-calen |
| 2631 | root     | 20 | 0  | 576712  | 22340  | 17016 | S | 0.0   | 1.1  | 0:00.10 | unity-fallback- |
| 801  | root     | 20 | 0  | 294216  | 11936  | 7580  | S | 0.0   | 0.6  | 0:00.09 | polkitd         |
| 2243 | root     | 20 | 0  | 1179456 | 25764  | 19856 | S | 0.0   | 1.3  | 0:00.09 | evolution-sourc |
| 2595 | root     | 20 | 0  | 431444  | 22052  | 16736 | S | 0.0   | 1.1  | 0:00.09 | polkit-gnome-au |
| 1834 | root     | 20 | 0  | 173608  | 732    | 456   | S | 0.0   | 0.0  | 0:00.08 | gpg-agent       |
| 1906 | root     | 20 | 0  | 553240  | 14908  | 12836 | S | 0.0   | 0.7  | 0:00.08 | gnome-session-b |
| 1979 | root     | 20 | 0  | 1377892 | 21668  | 14856 | S | 0.0   | 1.1  | 0:00.08 | indicator-datet |
| 1987 | root     | 20 | 0  | 549648  | 25540  | 19540 | S | 0.0   | 1.2  | 0:00.08 | indicator-print |
| 831  | root     | 20 | 0  | 19472   | 240    | 0     | S | 0.0   | 0.0  | 0:00.07 | irqbalance      |
| 1855 | root     | 20 | 0  | 429900  | 21948  | 18580 | S | 0.0   | 1.1  | 0:00.07 | ibus-x11        |
| 2813 | root     | 20 | 0  | 515316  | 18236  | 13944 | S | 0.0   | 0.9  | 0:00.07 | zeitgeist-datah |
| 23   | root     | 20 | 0  | 0       | 0      | 0     | S | 0.0   | 0.0  | 0:00.06 | ksoftirqd/3     |
| 1808 | root     | 20 | 0  | 39864   | 316    | 12    | S | 0.0   | 0.0  | 0:00.06 | upstart-dbus-br |
| 1872 | root     | 20 | 0  | 206972  | 6492   | 5848  | S | 0.0   | 0.3  | 0:00.06 | at-spi2-registr |


```

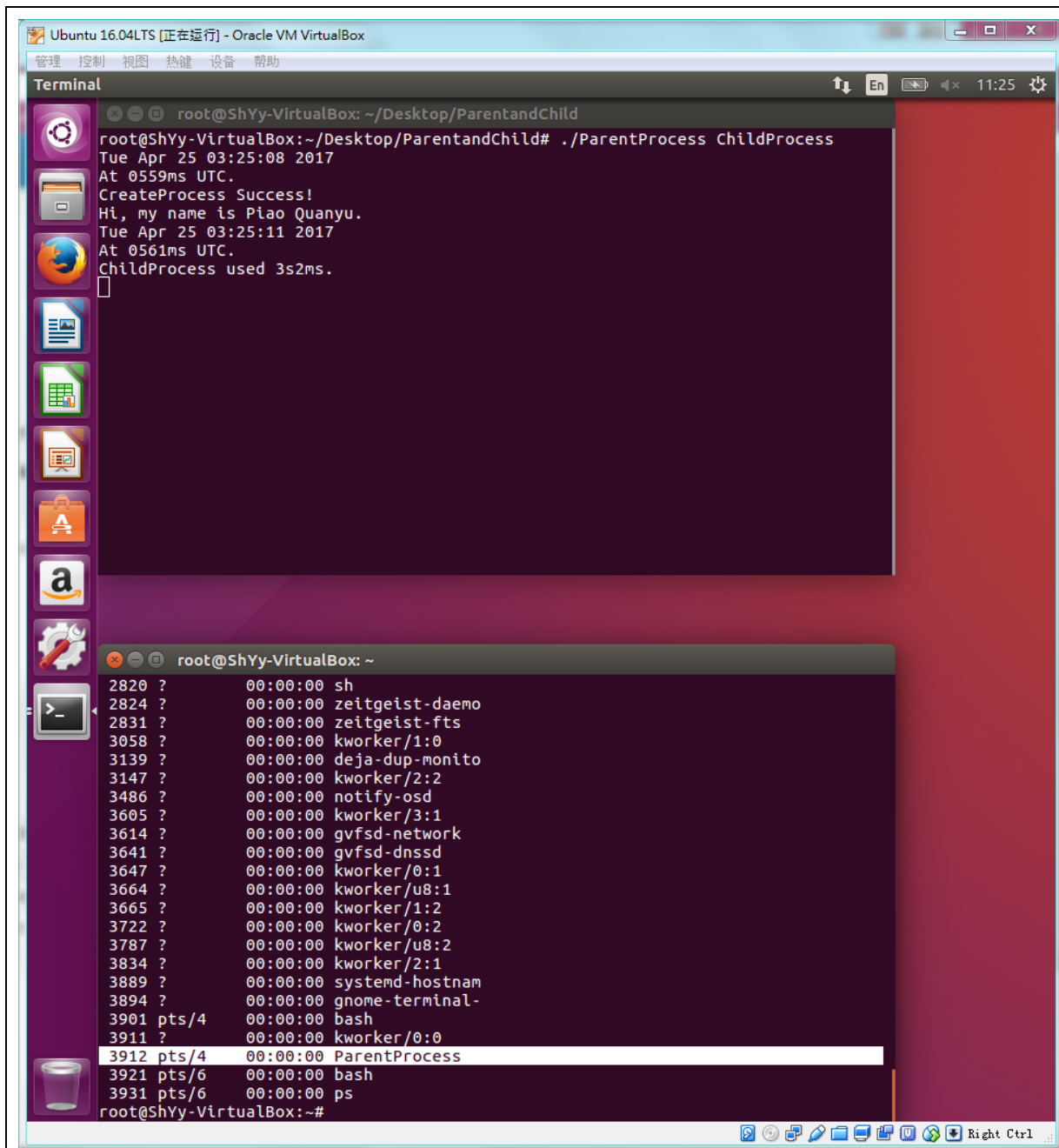
(图 2) top 命令，T（按进程时间累计）排序

```
top - 11:23:50 up 1:14, 1 user, load average: 0.11, 0.07, 0.06
Tasks: 198 total, 2 running, 196 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.6 us, 0.5 sy, 0.0 ni, 98.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2048128 total, 484204 free, 611404 used, 952520 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 1371484 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 2624 root        20   0 1507904 120676 27064 S   0.0   5.9   0:02.97 gnome-software
   921 root        20   0 457432 103424 41916 S   1.2   5.0   0:25.34 Xorg
 1918 root        20   0 1244620 87420 52532 S   1.9   4.3   0:28.14 compiz
 2688 root        20   0 869356 62292 22472 S   0.0   3.0   0:00.22 evolution-calen
 2582 root        20   0 974652 58552 43116 S   0.0   2.9   0:03.85 nautilus
 2736 root        20   0 1076004 52524 14132 S   0.0   2.6   0:00.12 evolution-calen
 2724 root        20   0 829480 51364 15084 S   0.0   2.5   0:00.21 evolution-calen
 1915 root        20   0 636388 41396 31464 S   0.0   2.0   0:00.60 hud-service
 3815 root        20   0 663240 36600 28128 R   2.5   1.8   0:00.86 gnome-terminal-
 2691 root        20   0 635456 36488 10792 S   0.0   1.8   0:00.62 fwupd
 2587 root        20   0 590972 34336 27228 S   0.0   1.7   0:00.17 nm-applet
 1908 root        20   0 564972 32816 25528 S   0.0   1.6   0:00.23 unity-panel-ser
 1892 root        20   0 864252 31460 24512 S   0.0   1.5   0:00.36 unity-settings-
 1893 root        20   0 525604 29796 22088 S   0.0   1.5   0:00.86 bamfdamon
 1852 root        20   0 479432 29728 24708 S   0.0   1.5   0:00.28 ibus-ui-gtk3
 3486 root        20   0 449116 28868 22448 S   0.0   1.4   0:00.14 notify-osd
 1982 root        20   0 651040 26636 21472 S   0.0   1.3   0:00.13 indicator-keybo
 2243 root        20   0 1179456 25764 19856 S   0.0   1.3   0:00.09 evolution-sourc
 1987 root        20   0 549648 25540 19540 S   0.0   1.2   0:00.08 indicator-print
 2755 root        20   0 779844 23484 16720 S   0.0   1.1   0:00.03 evolution-addre
 2631 root        20   0 576712 22340 17016 S   0.0   1.1   0:00.10 unity-fallback-
 2595 root        20   0 431444 22052 16736 S   0.0   1.1   0:00.09 polkit-gnome-au
 1855 root        20   0 429900 21948 18580 S   0.0   1.1   0:00.07 ibus-x11
 1979 root        20   0 1377892 21668 14856 S   0.0   1.1   0:00.08 indicator-datet
 2734 root        20   0 704372 19812 17032 S   0.0   1.0   0:00.04 evolution-addre
 2813 root        20   0 515316 18236 13944 S   0.0   0.9   0:00.07 zeitgeist-datah
   680 root        20   0 462524 16972 14512 S   0.0   0.8   0:00.12 NetworkManager
 2831 root        20   0 321400 15116 12748 S   0.0   0.7   0:00.03 zeitgeist-fts
 1906 root        20   0 553240 14908 12836 S   0.0   0.7   0:00.08 gnome-session-b
   586 whoopsie    20   0 378420 14588 10860 S   0.0   0.7   0:00.04 whoopsie
 1445 colord      20   0 320644 13164 10356 S   0.0   0.6   0:00.12 colord
 2167 root        20   0 395940 12440 11012 S   0.0   0.6   0:00.02 indicator-appli
 2646 root        20   0 382904 12312 7908 S   0.0   0.6   0:00.03 udisksd
 2640 root        20   0 377160 12276 8788 S   0.0   0.6   0:00.02 gvfs-udisks2-vo
 1986 root        20   0 625576 12252 10808 S   0.0   0.6   0:00.03 indicator-sound
   801 root        20   0 294216 11936 7580 S   0.0   0.6   0:00.09 polkitd
 2824 root        20   0 423428 11004 7792 S   0.0   0.5   0:00.02 zeitgeist-daemo
 1851 root        20   0 284684 10364 7548 S   0.0   0.5   0:00.01 ibus-dconf
 1778 root        20   0 93420 10352 9656 S   0.0   0.5   0:00.03 window-stack-br
 1432 root        20   0 354220 10212 8932 S   0.0   0.5   0:00.05 upowerd
 1978 root        20   0 366584 10076 8448 S   0.0   0.5   0:00.02 indicator-power
   874 root        20   0 274828 9508 8280 S   0.0   0.5   0:00.02 cups-browsed
 2672 root        20   0 410684 9044 8092 S   0.0   0.4   0:00.00 gvfs-afc-volume
 2703 root        20   0 370744 8992 7956 S   0.0   0.4   0:00.02 gvfsd-trash
 1976 root        20   0 367268 8960 8056 S   0.0   0.4   0:00.02 indicator-messa
 3139 root        20   0 448372 8760 7860 S   0.0   0.4   0:00.01 deja-dup-monito
 3614 root        20   0 444920 8728 7740 S   0.0   0.4   0:00.01 gvfsd-network
 1988 root        20   0 643392 8696 7688 S   0.0   0.4   0:00.02 indicator-sessi
```

(图 3) top 命令，M（按内存占用率）排序

操作系统课程设计实验报告



(图 4) `ps -A` 命令，查看所有进程，获取到 ParentProcess 进程的 PID

Ubuntu 16.04LTS [正在运行] - Oracle VM VirtualBox

管理 控制 视图 热键 设备 帮助

Terminal

root@ShYy-VirtualBox: ~/Desktop/ParentandChild

root@ShYy-VirtualBox: ~

```
2820 ?      00:00:00 sh
2824 ?      00:00:00 zeitgeist-daemo
2831 ?      00:00:00 zeitgeist-fts
3058 ?      00:00:00 kworker/1:0
3139 ?      00:00:00 deja-dup-monito
3147 ?      00:00:00 kworker/2:2
3486 ?      00:00:00 notify-osd
3605 ?      00:00:00 kworker/3:1
3614 ?      00:00:00 gvfsd-network
3641 ?      00:00:00 gvfsd-dnssd
3647 ?      00:00:00 kworker/0:1
3664 ?      00:00:00 kworker/u8:1
3665 ?      00:00:00 kworker/1:2
3722 ?      00:00:00 kworker/0:2
3787 ?      00:00:00 kworker/u8:2
3834 ?      00:00:00 kworker/2:1
3889 ?      00:00:00 systemd-hostnam
3894 ?      00:00:00 gnome-terminal-
3901 pts/4    00:00:00 bash
3911 ?      00:00:00 kworker/0:0
3912 pts/4    00:00:00 ParentProcess
3921 pts/6    00:00:00 bash
3931 pts/6    00:00:00 ps
root@ShYy-VirtualBox:~#
```

root@ShYy-VirtualBox: ~

```
top - 11:26:12 up 1:16, 1 user, load average: 0.16, 0.10, 0.06
Tasks: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.3 us, 0.3 sy, 0.0 ni, 99.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 2048128 total, 495564 free, 598488 used, 954076 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 1384284 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3912	root	20	0	4356	644	568	S	0.0	0.0	0:00.00	ParentProc+

(图 5) top -p PID 命令，查看 ParentProcess 进程的情况

The screenshot shows a terminal window titled "Ubuntu 16.04 LTS [正在运行] - Oracle VM VirtualBox". The user is at the prompt "root@ShYy-VirtualBox: ~/Desktop/ParentandChild".

First, the user runs `ps`, showing a list of processes. The process `ParentProcess` (PID 3912) is highlighted.

Next, the user runs `top -p 3912`. The output shows system statistics and a table of processes. The `ParentProc+` process is listed with PID 3912, user root, priority 20, 4356 K virtual memory, and 644 K resident memory.

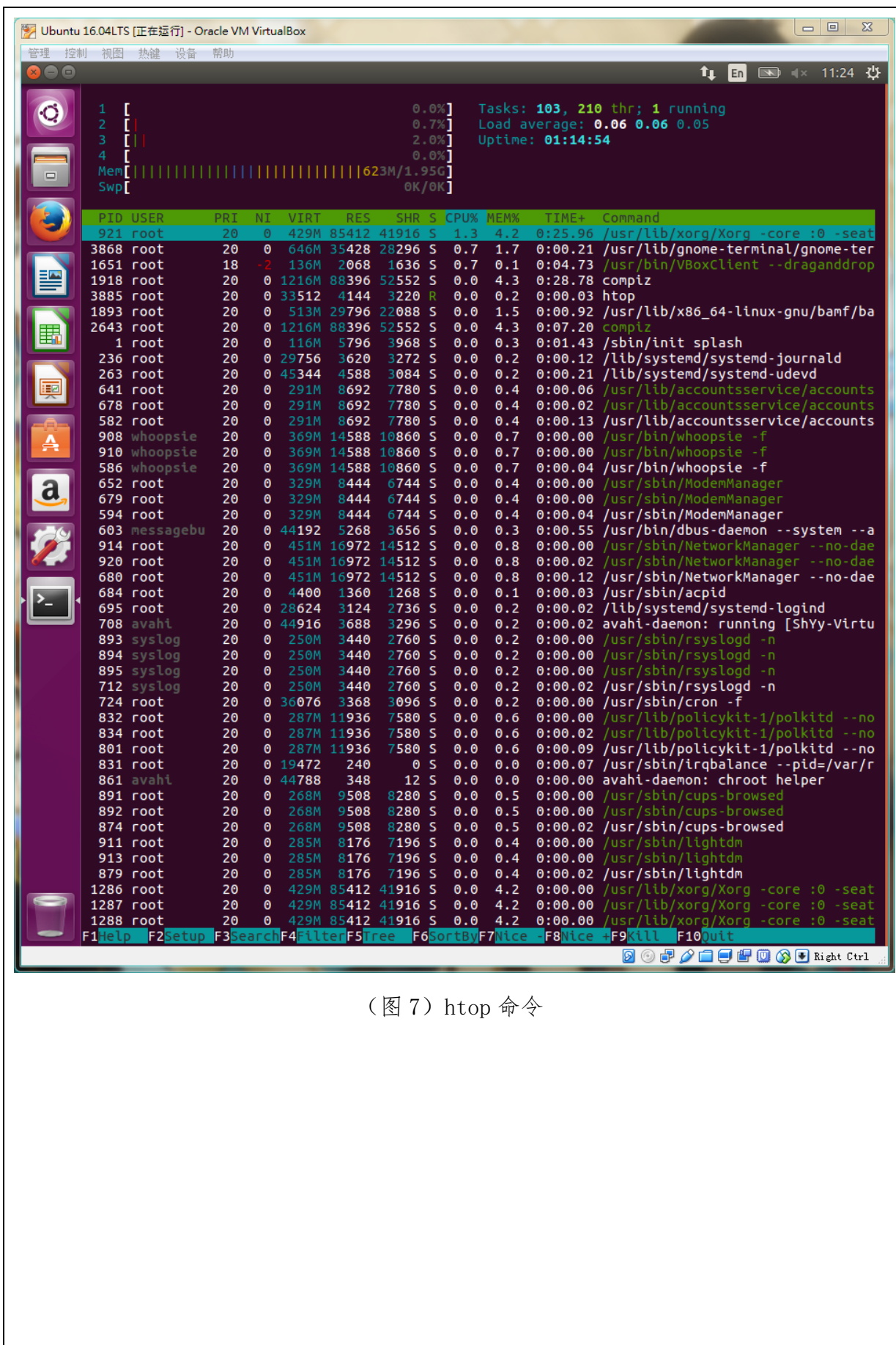
Finally, the user runs `pmap -d 3912`. The output shows the memory map for the `ParentProcess` child process. It lists memory addresses, sizes, permissions, offsets, devices, and mappings. The mappings include `ParentProcess`, `libc-2.23.so`, `ld-2.23.so`, `anon`, and `stack`.

```
root@ShYy-VirtualBox: ~/Desktop/ParentandChild
root@ShYy-VirtualBox: ~
3834 ?      00:00:00 kworker/2:1
3889 ?      00:00:00 systemd-hostnam
3894 ?      00:00:00 gnome-terminal-
3901 pts/4    00:00:00 bash
3911 ?      00:00:00 kworker/0:0
3912 pts/4    00:00:00 ParentProcess
3921 pts/6    00:00:00 bash
3931 pts/6    00:00:00 ps
root@ShYy-VirtualBox:~#
root@ShYy-VirtualBox:~# top -p 3912
top - 11:27:15 up 1:17, 1 user, load average: 0.11, 0.11, 0.07
Tasks: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.9 us, 1.1 sy, 0.0 ni, 98.0 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st
KiB Mem : 2048128 total, 490324 free, 603696 used, 954108 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 1379060 avail Mem

  PID USER      PR  NI   VIRT    RES    SHR S  %CPU  %MEM    TIME+  COMMAND
 3912 root        20   0   4356     644    568 S   0.0   0.0   0:00.00 ParentProc+
root@ShYy-VirtualBox:~# pmap -d 3912
3912: ./ParentProcess ChildProcess
Address            Kbytes Mode Offset                Device      Mapping
00000000000400000 4 r-x-- 00000000000000000 008:00001 ParentProcess
00000000000600000 4 r--- 00000000000000000 008:00001 ParentProcess
00000000000601000 4 rw-- 00000000000001000 008:00001 ParentProcess
00000000000891000 132 rw-- 00000000000000000 000:00000 [ anon ]
00007f23de64a000 1792 r-x-- 00000000000000000 008:00001 libc-2.23.so
00007f23de80a000 2044 ---- 000000000001c0000 008:00001 libc-2.23.so
00007f23dea09000 16 r--- 000000000001bf000 008:00001 libc-2.23.so
00007f23dea0d000 8 rw-- 000000000001c3000 008:00001 libc-2.23.so
00007f23dea0f000 16 rw-- 00000000000000000 000:00000 [ anon ]
00007f23dea13000 152 r-x-- 00000000000000000 008:00001 ld-2.23.so
00007f23dec1d000 12 rw-- 00000000000000000 000:00000 [ anon ]
00007f23dec36000 8 rw-- 00000000000000000 000:00000 [ anon ]
00007f23dec38000 4 r--- 00000000000025000 008:00001 ld-2.23.so
00007f23dec39000 4 rw-- 00000000000026000 008:00001 ld-2.23.so
00007f23dec3a000 4 rw-- 00000000000000000 000:00000 [ anon ]
00007ffd3e71e000 132 rw-- 00000000000000000 000:00000 [ stack ]
00007ffd3e76b000 8 r--- 00000000000000000 000:00000 [ anon ]
00007ffd3e76d000 8 r-x-- 00000000000000000 000:00000 [ anon ]
fffffffff600000 4 r-x-- 00000000000000000 000:00000 [ anon ]
mapped: 4356K writeable/private: 320K shared: 0K
root@ShYy-VirtualBox:~#
```

(图 6) pmap -d PID 命令，查看 ParentProcess 进程的内存使用情况

操作系统课程设计实验报告



(图 7) htop 命令

六、 讨论、心得

此次实验，模拟了我们在 Windows 下经常使用的任务管理器，了解了操作系统对进程信息的控制，对进程的操作。对操作系统的内存管理、地址空间布局等概念有了进一步认识。

值得注意的是，此次实验在 Linux 下并没有编码，但是通过 Terminal 以及相应的系统命令我们同样完成了与 Windows 下类似的操作。而且通过简要对比 top 和 htop 命令，可以发现 htop 比 top 要更加友好、易用。