

北京理工大学

操作系统课程设计

实验四、程序控制	实验四，内存监控
----------	----------

学院：计算机学院
专业：计算机科学与技术
学生姓名：夏奇拉
学号：1820171025
：07111705

目录

目的.....	3
问题讨论.....	3
执行 [Windows]	5
结果与分析 [Windows]	9
执行 [Linux]	10
结果与分析[Linux]	15
参考：	16

目的

Windows 实验：

Windows 设计了一个内存监视器，它需要：

实时显示当前系统的内存使用情况，包括系统地址空间的布局 and 物理内存的使用情况；

实时显示实验 2 进程控制（ParentProcess.exe）的虚拟地址空间布局和工作集信息

相关系统调用：

GetSystemInfo、VirtualQueryEx、VirtualAlloc、GetPerformanceInfo、GlobalMemoryStatusEx...

Linux 实验：

使用 top 命令查看系统，子命令 P、T、M

使用 ps -A 查看所有进程，找到 ProcessParent 的 pid

使用 top -p pid 查看 ProcessParent 程序的状态；

使用 pmap -d pid 查看 ProcessParent 的内存使用情况

问题讨论

实验要求程序必须显示

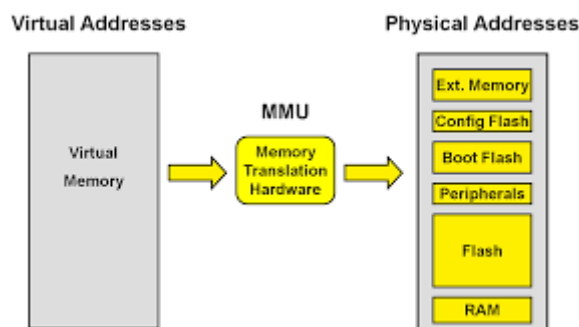
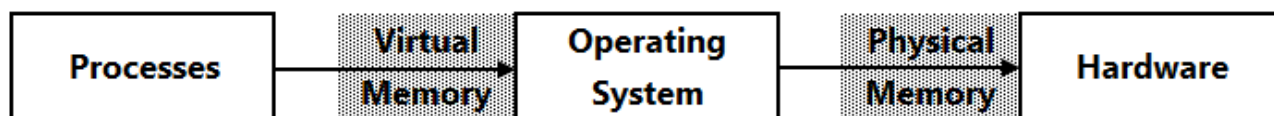
1. 系统的内存使用
 1. 系统地址空间布局
 2. 物理内存使用
2. 虚拟地址空间布局
3. parentprocess.exe 来自实验室 2 的工作信息

什么是系统内存？

计算机系统内存由称为随机存取存储器 (RAM) 的物理内存和虚拟内存组成。系统内存不是永久存储，就像计算机关闭时保存其内容的硬盘驱动器一样。

进程是包含在程序中的一组任务，它通过一系列称为线程的指令执行。进程在操作系统上运行，操作系统管理所有运行进程的硬件资源。

操作系统为在物理内存上运行的所有进程提供虚拟内存。操作系统的虚拟内存管理器使用一种称为分页的方法将虚拟地址空间映射到物理地址空间，这样所有进程都可以在物理内存上运行。



执行 [视窗]

获取系统信息

GetSystemInfo() 检索有关当前系统的信息。它接受指向接收信息的 **SYSTEM_INFO** 结构的指针。

SYSTEM_INFO 结构包含有关当前计算机系统的信息。

- wProcessorArchitecture：已安装操作系统的处理器架构。
- DwPageSize：页面大小和页面保护和提交的粒度。这是 **VirtualAlloc** 函数使用的页面大小。
- lpMinimumApplicationAddress：指向应用程序和动态链接库 (DLL) 可访问的最低内存地址的指针。
- lpMaximumApplicationAddress：指向应用程序和 DLL 可访问的最高内存地址的指针。
- dwAllocationGranularity：可以分配虚拟内存的起始地址的粒度。

虚拟查询 Ex

VirtualQueryEx 检索有关指定进程的虚拟地址空间内一系列连续页面的信息。返回值是信息缓冲区中返回的实际字节数。所有页面的可能状态包括 MEM_COMMIT、MEM_RESERV

E、MEM_FREE、MEM_PRIVATE、MEM_MAPPED 或 MEM_IMAGE。如果函数失败，则返回值为零。它采用以下参数：

- HANDLE hProcess：查询内存信息的进程句柄。
- LPCVOID lpAddress：指向要查询的页面区域的基地址的指针。 **GetSystemInfo** 函数用于确定主机上页面的大小。
- PMEMORY_BASIC_INFORMATION lpBuffer：指向 MEMORY_BASIC_INFORMATION 结构的指针，其中返回有关指定页面范围的信息。
- SIZE_T dwLength：lpBuffer 参数指向的缓冲区的大小，以字节为单位。

MEMORY_BASIC_INFORMATION 结构包含有关进程虚拟地址空间中页面范围的信息。其成员包括：

- PVOID BaseAddress：指向页面区域的基地址
- PVOID AllocationBase：指向由 **VirtualAlloc** 函数分配的页面范围的基地址的指针。BaseAddress 成员指向的页面包含在此分配范围内。
- DWORD AllocationProtect：最初分配区域时的内存保护选项。
- WORD 分区 ID
- SIZE_T RegionSize：从所有页面具有相同属性的基地址开始的区域大小，以字节为单位
- DWORD State：该区域中页面的状态。状态包括 MEM_COMMIT、MEM_FREE 和 MEM_RESERVE。
- DWORD Protect：区域内页面的访问保护
- DWORD Type：区域中页面的类型，包括 MEM_IMAGE、MEM_MAPPED 和 MEM_PRIVATE。

虚拟分配

VirtualAlloc 函数保留、提交或更改调用进程的虚拟地址空间中页面区域的状态。此函数分配的内存自动初始化为零。如果函数成功，返回值是页面分配区域的基地址。

每个页面都有一个关联的页面状态（空闲、保留或提交）。 **VirtualAlloc** 函数可以执行以下操作：

- 提交保留页面区域
- 保留一个免费页面区域

- 同时保留和提交空闲页面区域

它需要 4 个参数。

- LPVOID LpAddress：要分配的区域起始地址。
- SIZE_T dwSize：区域大小，以字节为单位
- DWORD flAllocationType：内存分配的类型。此参数必须包含以下值之一：MEM_COMMIT、MEM_RESERVE、MEM_RESET、MEM_RESET_UNDO
- DWORD flProtect：要分配的页面区域的内存保护。

如何使用此功能的示例：[Reserving and Committing Memory - Win32 apps | 微软学习](#)

获取性能信息

GetPerformanceInfo 函数保留包含在 **PERFORMANCE_INFORMATION** 结构中的性能值。

如果函数成功，则返回值为 TRUE。如果函数失败，则返回值为 FALSE。它采用以下参数：

- **PERFORMANCE_INFORMATION** pPerformanceInformation：指向接收性能信息的 **PERFORMANCE_INFORMATION** 结构的指针。
- **DWORD** cb： **PERFORMANCE_INFORMATION** 结构的大小，以字节为单位。

PERFORMANCE_INFORMATION 结构包含性能信息。其成员包括：

- **DWORD** cb：此结构的大小（以字节为单位）
- **SIZE_T** CommitTotal：系统当前提交的页数。
- **SIZE_T** CommitLimit：在不扩展页面文件的情况下，系统可以提交的当前最大页数。
- **SIZE_T** 提交峰值；自上次系统重启后同时处于提交状态的最大页数。
- **SIZE_T** PhysicalTotal：实际物理内存量，以页为单位。
- **SIZE_T** PhysicalAvailable：当前可用的物理内存量，以页为单位。
- **SIZE_T** SystemCache：系统缓存内存量，以页为单位。这是备用列表加上系统工作集的大小。
- **SIZE_T** KernelTotal：当前在分页和非分页内核池中的内存总和，以页为单位。
- **SIZE_T** KernelPaged：当前在分页内核池中的内存，以页为单位。
- **SIZE_T** KernelNopaged：当前在非分页内核池中的内存，以页为单位。
- **SIZE_T** PageSize：页面的大小，以字节为单位。

- DWORD 句柄数：
- 当前打开的句柄数。
- DWORD ProcessCount：当前进程数。
- DWORD ThreadCount：当前线程数。

全局内存状态 Ex

GlobalMemoryStatusEx 函数检索有关系统当前物理和虚拟内存使用情况的信息。如果函数成功，则返回值为非零。它接受一个参数：

- LPMEMORYSTATUSEX lpBuffer：指向接收有关内存可用性信息的 **MEMORYSTATUSEX** 结构的指针。

MEMORYSTATUSEX 结构包含有关物理和虚拟内存（包括扩展内存）的当前状态的信息。其成员包括：

- DWORD dwLength：结构的大小（以字节为单位）。该成员必须在调用 **GlobalMemoryStatusEx** 之前设置
- DWORD dwMemoryLoad：一个介于 0 和 100 之间的数字，指定正在使用的物理内存的近似百分比（0 表示没有内存使用，100 表示完全使用内存）。
- DWORDLONG ullTotalPhys：以字节为单位的实际物理内存量。
- DWORDLONG ullAvailPhys：当前可用的物理内存量，以字节为单位。它是备用列表、空闲列表和零列表的大小之和。
- DWORDLONG ullTotalPageFile：系统或当前进程的当前提交内存限制，以字节为单位，以较小者为准。要获取系统范围内提交的内存限制，请调用 **GetPerformanceInfo**
- DWORDLONG ullAvailPageFile：当前进程可以提交的最大内存量，以字节为单位。该值等于或小于系统范围内的可用提交值。要计算系统范围内的可用提交值，请调用 **GetPerformanceInfo** 并从 CommitLimit 的值中减去 **CommitTotal** 的值。
- DWORDLONG ullTotalVirtual：调用进程虚拟地址空间的用户模式部分的大小，以字节为单位
- DWORDLONG ullAvailVirtual：当前在调用进程的虚拟地址空间的用户模式部分中未保留和未提交的内存量，以字节为单位。
- DWORDLONG ullAvailExtendedVirtual：保留。该值始终为 0。

```
#include <windows.h>
#include <stdio.h>
#include <tchar.h>
#include <进程.h>
#include <Psapi.h>
#pragma comment(lib, "user32.lib")

// 用于将字节转换为 KB
#define DIV 1024

// 指定打印数字的字段宽度。
// 格式说明符 "%*I64d" 中的星号取一个整数
// 参数并使用它来填充和右对齐数字。
#define 宽度 7

无效打印系统信息 ()
{
printf("\n 系统信息");
printf("\n-----\n");

SYSTEM_INFO siSysInfo;

// 将硬件信息复制到 SYSTEM_INFO 结构中。

GetSystemInfo(&siSysInfo);

// 显示 SYSTEM_INFO 结构的内容。

printf("OEM ID: %u\n", siSysInfo.dwOemId);
printf("处理器数量: %u\n",
siSysInfo.dwNumberOfProcessors);
printf("页面大小: %u\n", siSysInfo.dwPageSize);
printf("处理器类型: %u\n", siSysInfo.dwProcessorType);
printf("最小应用地址: %lx\n",
siSysInfo.lpMinimumApplicationAddress);
printf("最大应用地址: %lx\n",
siSysInfo.lpMaximumApplicationAddress);
printf(" 活动处理器掩码: %u\n",
siSysInfo.dwActiveProcessorMask);
}

无效 PrintMemoryInfo()
{
printf("\n 内存信息\n");
```



```

printf("-----\n");

MEMORYSTATUSEX ms = {sizeof(MEMORYSTATUSEX)};

ms.dwLength = sizeof(ms);

// 检索有关系统当前物理内存使用情况的信息

GlobalMemoryStatusEx(&ms);

printf("使用的总内存: %ld%%\n", ms.dwMemoryLoad);

printf("\n 物理内存总量: %8.2I64fMB\n 可用物理内存: %8.2I64fMB\n 已用物理内存: %8.2I64fMB\n\n", ms.ullTotalPhys / (1024 * 1024.0), ms.ullAvailPhys / (1024 * 1024.0), ms.ullTotalPhys / (1024 * 1024.0) - ms.ullAvailPhys / (1024 * 1024.0));

printf("总虚拟内存: %8.2I64fMB\n 可用虚拟内存: %8.2I64fMB\n 已用虚拟内存: %8.2I64fMB\n\n", ms.ullTotalVirtual / (1024 * 1024.0), ms.ullAvailVirtual / (1024 * 1024.0), ms.ullTotalVirtual / (1024 * 1024.0) - ms.ullAvailVirtual / (1024 * 1024.0));
}

无效 PrintPerformanceInfo()
{
printf("\n 性能信息\n");
printf("-----\n");

PERFORMANCE_INFORMATION siPerfInfo;

// 将硬件信息复制到 SYSTEM_INFO 结构中。

GetPerformanceInfo(&siPerfInfo, siPerfInfo.cb);

printf("提交总数\t\t: %d 页\n", siPerfInfo.CommitTotal);
printf("提交限制\t\t: %d 页\n", siPerfInfo.CommitLimit);
printf("提交峰值\t\t: %d 页\n", siPerfInfo.CommitPeak);
printf("纸本总计\t\t: %d 页\n", siPerfInfo.PhysicalTotal);
printf("物理可用\t\t: %d 页\n", siPerfInfo.PhysicalAvailable);
printf("系统缓存\t\t: %d 页\n", siPerfInfo.SystemCache);
printf("内核总数\t\t: %d 页\n", siPerfInfo.KernelTotal);
printf("内核分页\t\t: %d 页\n", siPerfInfo.KernelPaged);
printf("Kernel Nonpaged\t\t: %d pages\n", siPerfInfo.KernelNonpaged);
printf("页面大小\t\t: %d MB\n", siPerfInfo.PageSize / (1024 * 1024.0));
printf("句柄计数\t\t: %d 个句柄\n", siPerfInfo.HandleCount);

```

```

printf("进程数\t\t: %d 进程\n", siPerfInfo.ProcessCount);
printf("线程计数\t\t: %d 线程\n", siPerfInfo.ThreadCount);
}

void PrintRunningProcesses (无效)
{
printf("\n 当前正在运行的进程\n");
printf("-----\n");

system("tasklist /FI \"IMAGENAME eq parentprocess.exe\"");
system("tasklist /FI \"IMAGENAME eq childprocess.exe\"");
system("tasklist /FI \"IMAGENAME eq memorymonitoring.exe\"");
}

void _tmain(int argc, TCHAR *argv[])
{

printf("实验 4: 内存监控\n");

打印系统信息();
打印内存信息 ();
打印性能信息 ();

启动信息 si;
PROCESS_INFORMATION pi;
ZeroMemory(&si, sizeof(si));
si.cb = sizeof(si);
ZeroMemory(&pi, sizeof(pi));
if (argc != 3) /* argc 应该是 2 才能正确执行 */
{
printf("内存监控中的使用情况: %s [cmdline]\n", argv[0]);
返回;
}
// 启动子进程。
if (!CreateProcess(NULL, // 没有模块名称 (使用命令行)
argv[1], // 命令行
NULL, // 进程句柄不可继承
NULL, // 线程句柄不可继承
FALSE, // 将句柄继承设置为 FALSE
0, // 没有创建标志
NULL, // 使用父级的环境块
NULL, // 使用父目录的起始目录
&si, // 指向 STARTUPINFO 结构的指针

```

```
&pi) // 指向 PROCESS_INFORMATION 结构的指针
)
{
printf("CreateProcess 失败 (%d).\n", GetLastError());
返回;
}

PrintRunningProcesses();

// 只询问任务列表命令

睡眠 (5000) ;

返回 0;
}
```

结果与分析 [Windows]

```
C:\> Developer Command Prompt for VS 2022

LAB 4: MEMORY MONITORING

SYSTEM INFORMATION
-----
OEM ID: 0
Number of processors: 2
Page size: 4096
Processor type: 586
Minimum application address: 10000
Maximum application address: 7ffefffff
Active processor mask: 3

MEMORY INFORMATION
-----
Total memory in use: 76%

Total Physical Memory      : 3982.88MB
Available Physical Memory  : 926.84MB
Used Physical Memory       : 3056.04MB

Total Virtual Memory       : 2047.88MB
Available Virtual Memory   : 2036.41MB
Used Virtual Memory        : 11.47MB

PERFORMANCE INFORMATION
-----
Commit Total      : 1005262 pages
Commit Limit      : 1772187 pages
Commit Peak       : 1534224 pages
Physical Total     : 1019618 pages
Physical Available : 237269 pages
System Cache       : 227442 pages
Kernel Total       : 107746 pages
Kernel Paged       : 70820 pages
Kernel Nonpaged    : 36926 pages
Page Size          : 0 MB
Handle Count       : 73472 handles
Process Count      : 182 processes
Thread Count       : 1625 threads

RUNNING PARENT PROCESS
-----
The child process start time is: 20h:50m:15s.09ms
Hello my name is Xiaqila
Delay 3s

Image Name          PID Session Name      Session#  Mem Usage
=====
parentprocess.exe   4708 Console          4         3,700 K

Image Name          PID Session Name      Session#  Mem Usage
=====
childprocess.exe    13752 Console         4         3,356 K
The child process end time is: 20h:50m:18s.71ms
The child process elapsed time is: 03s.62ms

C:\Users\likkl\BIT-100073007-Operating-Systems-Course\Lab-4-Memory-Monitoring\sourcecode\Windows>
```

执行 [Linux]

top（进程表）命令显示了 Linux 中正在运行的进程的实时视图。默认情况下，它按 %CPU 列对进程列表进行排序。以下命令可用于使用不同的列进行排序：

- P. 按 %CPU 列排序
- T. 按 TIME+ 列排序

- M. 按 %MEM 列排序

使用 P 子命令按 %CPU 排序

```
shak@ubuntu-machine: ~
top - 13:26:06 up 20:30, 1 user, load average: 0.98, 1.01, 0.96
Tasks: 211 total, 1 running, 210 sleeping, 0 stopped, 0 zombie
%Cpu(s): 15.1 us, 4.7 sy, 0.0 ni, 80.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3815.5 total, 251.3 free, 1593.6 used, 1970.6 buff/cache
MiB Swap: 7629.0 total, 7585.5 free, 43.5 used. 1632.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
14853	shak	20	0	2494520	169180	96256	S	18.8	4.3	2:00.14	Isolated +
14195	shak	20	0	3219932	339844	165516	S	7.9	8.7	2:29.44	firefox
1873	shak	20	0	4524064	264408	104200	S	3.6	6.8	10:46.08	gnome-she+
12998	root	20	0	0	0	0	I	2.0	0.0	0:03.05	kworker/u+
2858	shak	20	0	1424780	81248	43392	S	1.3	2.1	1:44.01	nautilus
3786	shak	20	0	363564	68836	40652	S	1.3	1.8	1:12.82	Xwayland
15442	shak	20	0	21840	4128	3308	R	1.3	0.1	0:04.84	top
14825	root	20	0	0	0	0	I	1.0	0.0	0:01.75	kworker/u+
671	message+	20	0	11120	6600	3892	S	0.7	0.2	0:25.34	dbus-daem+
672	root	20	0	687576	18244	14800	S	0.7	0.5	0:27.33	NetworkMa+
12907	shak	20	0	566384	56120	41496	S	0.7	1.4	0:24.75	gnome-ter+
15523	root	20	0	0	0	0	I	0.7	0.0	0:00.40	kworker/1+
14	root	20	0	0	0	0	I	0.3	0.0	0:25.64	rcu_preem+
78	root	0	-20	0	0	0	I	0.3	0.0	0:13.15	kworker/u+
250	root	19	-1	65128	27872	26276	S	0.3	0.7	0:04.64	systemd-j+
608	systemd+	20	0	14824	6168	5376	S	0.3	0.2	2:17.87	systemd-o+
609	systemd+	20	0	25928	13896	9300	S	0.3	0.4	0:05.32	systemd-r+

按时间排序 + 使用 T 子命令

```
shak@ubuntu-machine: ~  
top - 13:27:33 up 20:31, 1 user, load average: 0.58, 0.87, 0.91  
Tasks: 210 total, 2 running, 208 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 17.1 us, 7.0 sy, 0.0 ni, 75.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB Mem : 3815.5 total, 215.1 free, 1624.6 used, 1975.8 buff/cache  
MiB Swap: 7629.0 total, 7585.5 free, 43.5 used. 1599.6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1873	shak	20	0	4525924	264600	104352	S	18.4	6.8	10:55.17	gnome-she+
14195	shak	20	0	3220012	342284	165516	S	1.0	8.8	2:31.21	firefox
608	systemd+	20	0	14824	6168	5376	R	0.3	0.2	2:18.25	systemd-o+
14853	shak	20	0	2494520	169408	96256	S	17.1	4.3	2:13.13	Isolated +
2858	shak	20	0	1424780	81792	43428	S	0.0	2.1	1:44.21	nautilus
3786	shak	20	0	363564	68836	40652	S	0.0	1.8	1:13.09	Xwayland
14149	shak	20	0	1079108	300592	147280	S	0.0	7.7	1:08.13	soffice.b+
1154	root	20	0	454832	49464	16888	S	0.0	1.3	0:58.35	packageki+
2019	shak	20	0	323464	11116	6464	S	0.0	0.3	0:35.22	ibus-daem+
684	root	20	0	948576	38592	19988	S	0.0	1.0	0:29.03	snappd
672	root	20	0	687576	18244	14800	S	0.0	0.5	0:27.37	NetworkMa+
2178	shak	20	0	1517972	238740	29592	S	0.0	6.1	0:26.79	snap-store
12907	shak	20	0	566384	56120	41496	S	5.9	1.4	0:25.81	gnome-ter+
14	root	20	0	0	0	0	I	0.3	0.0	0:25.75	rcu_preem+
671	message+	20	0	11120	6600	3892	S	0.0	0.2	0:25.38	dbus-daem+
669	avahi	20	0	7708	3928	3412	S	0.0	0.1	0:15.92	avahi-dae+
78	root	0	-20	0	0	0	I	0.7	0.0	0:13.32	kworker/u+

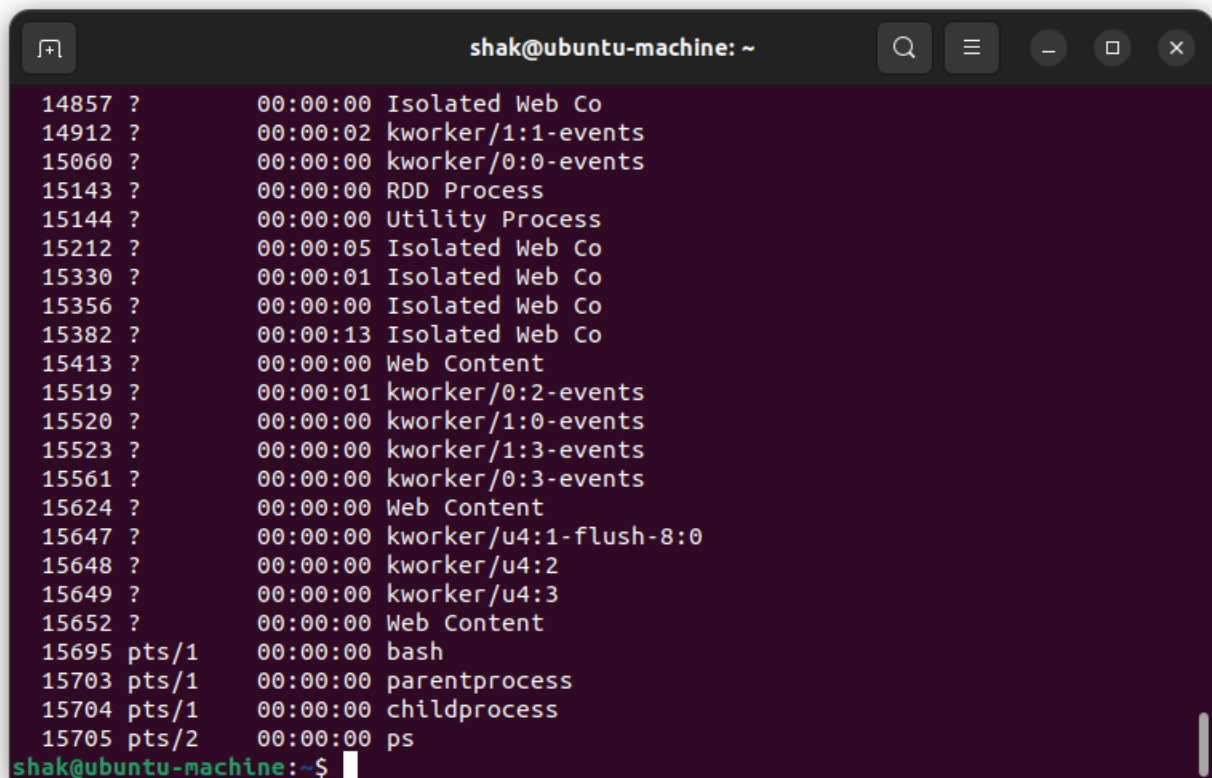
使用 M 子命令按 %MEM 排序

```
shak@ubuntu-machine: ~  
top - 13:27:51 up 20:32, 1 user, load average: 0.82, 0.91, 0.93  
Tasks: 210 total, 1 running, 209 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 9.0 us, 2.7 sy, 0.0 ni, 87.5 id, 0.8 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB Mem : 3815.5 total, 199.9 free, 1637.6 used, 1978.0 buff/cache  
MiB Swap: 7629.0 total, 7585.5 free, 43.5 used. 1585.2 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
14195	shak	20	0	3219928	341924	165516	S	0.0	8.8	2:31.30	firefox
14149	shak	20	0	1085192	306844	147288	S	0.0	7.9	1:11.29	soffice.b+
1873	shak	20	0	4527520	264792	104352	S	1.0	6.8	10:58.20	gnome-she+
2178	shak	20	0	1517972	238740	29592	S	0.0	6.1	0:26.79	snap-store
14853	shak	20	0	2494520	168980	96256	S	15.4	4.3	2:15.80	Isolated +
15212	shak	20	0	2467292	132384	87424	S	0.0	3.4	0:05.88	Isolated +
14340	shak	20	0	2435184	104376	78828	S	0.0	2.7	0:03.32	Privilege+
15330	shak	20	0	2448772	98504	82440	S	0.0	2.5	0:01.30	Isolated +
14613	shak	20	0	2431196	89020	68392	S	0.0	2.3	0:01.63	WebExtens+
15356	shak	20	0	2411388	82480	68892	S	0.0	2.1	0:00.53	Isolated +
2858	shak	20	0	1432976	81860	43488	S	0.0	2.1	1:44.32	nautilus
14857	shak	20	0	2410312	80860	67416	S	0.0	2.1	0:00.67	Isolated +
3786	shak	20	0	363524	68836	40652	S	0.0	1.8	1:13.10	Xwayland
15382	shak	20	0	2405544	62784	50848	S	0.0	1.6	0:00.29	Web Conte+
15409	shak	20	0	2405544	62268	50340	S	0.0	1.6	0:00.30	Web Conte+
15413	shak	20	0	2405540	61932	50076	S	0.0	1.6	0:00.28	Web Conte+
3793	shak	20	0	590804	61564	44512	S	0.0	1.6	0:00.58	gsd-xsett+

使用 ps -A 查找父进程的 PID

ps 命令允许您轻松列出系统上运行的进程的状态。



```
shak@ubuntu-machine: ~  
14857 ?      00:00:00 Isolated Web Co  
14912 ?      00:00:02 kworker/1:1-events  
15060 ?      00:00:00 kworker/0:0-events  
15143 ?      00:00:00 RDD Process  
15144 ?      00:00:00 Utility Process  
15212 ?      00:00:05 Isolated Web Co  
15330 ?      00:00:01 Isolated Web Co  
15356 ?      00:00:00 Isolated Web Co  
15382 ?      00:00:13 Isolated Web Co  
15413 ?      00:00:00 Web Content  
15519 ?      00:00:01 kworker/0:2-events  
15520 ?      00:00:00 kworker/1:0-events  
15523 ?      00:00:00 kworker/1:3-events  
15561 ?      00:00:00 kworker/0:3-events  
15624 ?      00:00:00 Web Content  
15647 ?      00:00:00 kworker/u4:1-flush-8:0  
15648 ?      00:00:00 kworker/u4:2  
15649 ?      00:00:00 kworker/u4:3  
15652 ?      00:00:00 Web Content  
15695 pts/1     00:00:00 bash  
15703 pts/1     00:00:00 parentprocess  
15704 pts/1     00:00:00 childprocess  
15705 pts/2     00:00:00 ps  
shak@ubuntu-machine:~$
```

parentprocess 的 pid – 15703。这在 parentprocess 命令的输出中得到确认


```
shak@ubuntu-machine: ~/Documents/BIT-100073007-Operat...
shak@ubuntu-machine:~/Documents/BIT-100073007-Operating-Systems-Course/Lab-2-Pro
cess-Control/source-code/Linux$ ./parentprocess childprocess
The child process start time is: 1671730209 seconds 522600 micro seconds
Hello my name is Xiqila
Delay 3s
The child process end time is: 1671730212 seconds 525338 micro seconds
The child process elapsed time is: 3s.2ms

Parent process PID: 15703
Child process PID: 15704
shak@ubuntu-machine:~/Documents/BIT-100073007-Operating-Systems-Course/Lab-2-Pro
cess-Control/source-code/Linux$
```

使用 top -p pid 检查父进程的状态

```
shak@ubuntu-machine: ~
top - 13:35:29 up 20:39, 1 user, load average: 0.65, 0.79, 0.88
Tasks: 1 total, 0 running, 1 sleeping, 0 stopped, 0 zombie
%Cpu(s): 9.4 us, 6.2 sy, 0.0 ni, 84.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 3815.5 total, 132.2 free, 1705.0 used, 1978.3 buff/cache
MiB Swap: 7629.0 total, 7556.2 free, 72.8 used. 1365.9 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 15823 shak      20   0   2772   1072   984  S   0.0   0.0   0:00.00 parentpro+

shak@ubuntu-machine:~$
```


使用 `pmap -d pid` 查看 ProcessParent 的内存使用情况的 `pmap` 命令用于显示进程的内存映射。

```
shak@ubuntu-machine:~$ pmap -d 15823
shak@ubuntu-machine:~$ pmap -d 15984
15984:  ./parentprocess childprocess
Address      Kbytes Mode  Offset          Device      Mapping
000055926e337000      4 r---- 0000000000000000 008:00008 parentprocess
000055926e338000      4 r-x-- 0000000000000100 008:00008 parentprocess
000055926e339000      4 r---- 0000000000000200 008:00008 parentprocess
000055926e33a000      4 r---- 0000000000000200 008:00008 parentprocess
000055926e33b000      4 rw--- 0000000000000300 008:00008 parentprocess
000055926e897000     132 rw--- 0000000000000000 000:00000 [ anon ]
00007fb6d1000000     160 r---- 0000000000000000 008:00009 libc.so.6
00007fb6d1028000    1620 r-x-- 0000000000002800 008:00009 libc.so.6
00007fb6d11bd000     352 r---- 000000000001bd00 008:00009 libc.so.6
00007fb6d1215000      16 r---- 0000000000021400 008:00009 libc.so.6
00007fb6d1219000       8 rw--- 0000000000021800 008:00009 libc.so.6
00007fb6d121b000      52 rw--- 0000000000000000 000:00000 [ anon ]
00007fb6d126b000      12 rw--- 0000000000000000 000:00000 [ anon ]
00007fb6d127d000       8 rw--- 0000000000000000 000:00000 [ anon ]
00007fb6d127f000       8 r---- 0000000000000000 008:00009 ld-linux-x86-64.so.2
00007fb6d1281000    168 r-x-- 0000000000002000 008:00009 ld-linux-x86-64.so.2
00007fb6d12ab000     44 r---- 000000000002c000 008:00009 ld-linux-x86-64.so.2
00007fb6d12b7000       8 r---- 0000000000037000 008:00009 ld-linux-x86-64.so.2
00007fb6d12b9000       8 rw--- 0000000000039000 008:00009 ld-linux-x86-64.so.2
00007ffda0bf9000    132 rw--- 0000000000000000 000:00000 [ stack ]
00007ffda0d32000      16 r---- 0000000000000000 000:00000 [ anon ]
00007ffda0d36000       8 r-x-- 0000000000000000 000:00000 [ anon ]
fffffffff6000000       4 --x-- 0000000000000000 000:00000 [ anon ]
mapped: 2776K    writeable/private: 356K    shared: 0K
shak@ubuntu-machine:~$
```

参考：

- <https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-getsysteminfo>
- https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/ns-sysinfoapi-system_info
- <https://learn.microsoft.com/en-gb/windows/win32/sysinfo/getting-hardware-information?redirectedfrom=MSDN>
- https://www.installsetupconfig.com/win32programming/windowsvolumeapis1_6.html
- <https://learn.microsoft.com/en-us/windows/win32/winprog64/virtual-address-space>
- <https://learn.microsoft.com/en-us/windows/win32/memory/memory-management>
- [#](https://www.tutorialspoint.com/operating_system/os_memory_management.htm)
- [随机存取存储器 \(RAM\) 如何影响性能 | 戴尔美国](#)
- [Windows 10 中的物理和虚拟内存 - Microsoft Community](#)
- [页面状态 - Win32 应用 | 微软学习](#)

- [MEMORY BASIC INFORMATION \(winnt.h\) - Win32 应用程序 | 微软学习](#)