

北京理工大学

操作系统课程设计

实验四、进程控制	Experiment 4, Memory Monitoring
----------	---------------------------------

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
班级：07111705

Table of Contents

Purpose.....3

Problem Discussion.....3

Execution [Windows].....5

 Results and Analysis [Windows].....9

Execution [Linux].....10

 Results and Analysis [Linux].....15

Reference:.....16

Purpose

Experiment for Windows:

Windows designs a memory monitor that requires:

Real-time display of the memory usage in the current system, including the layout of the system address space and the usage of physical memory;

Real-time display of the virtual address space layout and working set information of the experiment 2 process control (ParentProcess.exe)

Related syscalls:

GetSystemInfo, VirtualQueryEx, VirtualAlloc, GetPerformanceInfo, GlobalMemoryStatusEx...

Linux experiment:

Use the top command to view the system, subcommands P, T, M

Use ps -A to view all processes and find the pid of ProcessParent

Use top -p pid to check the status of the ProcessParent program;

Use pmap -d pid to view the memory usage of ProcessParent

Problem Discussion

The experiment requires that the program must display

1. the system's memory usage
 1. the system address space layout
 2. the physical memory usage
2. virtual address space layout
3. parentprocess.exe from lab 2's working information

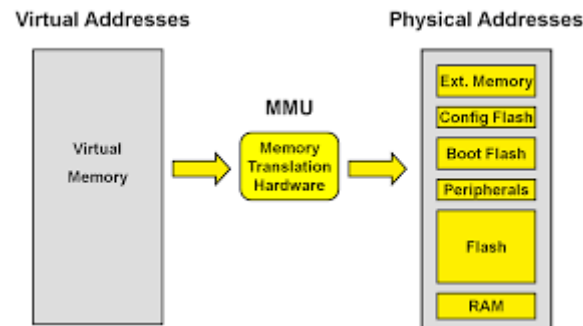
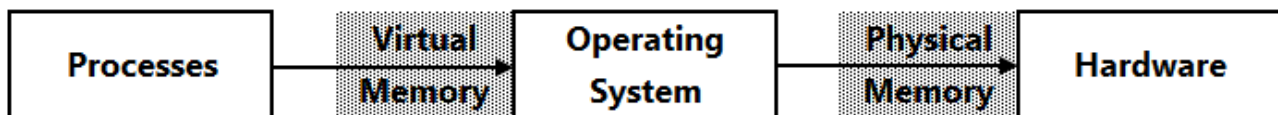
What is system memory?

A computer's system memory is made up of physical memory, called Random Access Memory (RAM), and virtual memory. System memory is not permanent storage, like a hard disk drive that saves its contents when the computer is switched off.

A process is a set of tasks contained in a program, which executes through a sequence of instructions called threads. Processes run on the operating system and the operating system manages the hardware resources for all the running processes.

The operating system provides virtual memory to all processes which run on physical memory. The virtual memory manager of the operating system applies a method called

Paging to map the virtual address space to the physical address space, in such a way that all processes can get to run on the physical memory.



Execution [Windows]

GetSystemInfo

GetSystemInfo() retrieves information about the current system. It accepts a pointer to a **SYSTEM_INFO** structure that receives the information.

The **SYSTEM_INFO** structure contains information about the current computer system.

- **wProcessorArchitecture**: The processor architecture of the installed operating system.
- **DwPageSize**: The page size and the granularity of page protection and commitment. This is the page size used by the **VirtualAlloc** function.
- **lpMinimumApplicationAddress**: A pointer to the lowest memory address accessible to applications and dynamic-link libraries (DLLs).
- **lpMaximumApplicationAddress**: A pointer to the highest memory address accessible to applications and DLLs.
- **dwAllocationGranularity**: The granularity for the starting address at which virtual memory can be allocated.

VirtualQueryEx

The **VirtualQueryEx** retrieves information about a range of consecutive pages within the virtual address space of specified processes. The return value is the actual number of bytes returned in the information buffer. The possible states of all the pages include **MEM_COMMIT**, **MEM_RESERVE**, **MEM_FREE**, **MEM_PRIVATE**, **MEM_MAPPED** or

MEM_IMAGE. If the function fails, the return value is zero. It takes the following parameters:

- HANDLE hProcess: A handle to the process whose memory information is queried.
- LPCVOID lpAddress: A pointer to the base address of the region of pages to be queried. The **GetSystemInfo** function is used to determine the size of a page on the host computer.
- PMEMORY_BASIC_INFORMATION lpBuffer: A pointer to a **MEMORY_BASIC_INFORMATION** structure in which information about the specified page range is returned.
- SIZE_T dwLength: The size of the buffer pointed to by the lpBuffer parameter in bytes.

The **MEMORY_BASIC_INFORMATION** structure contains information about a range of pages in the virtual address space of a process. Its members include:

- PVOID BaseAddress: A point to the base address of the region of pages
- PVOID AllocationBase: A pointer to the base address of a range of pages allocated by the **VirtualAlloc** function. The page pointed to by the BaseAddress member isb contained within this allocation range.
- DWORD AllocationProtect: The memory protection option when the region was initially allocated.
- WORD PartitionId
- SIZE_T RegionSize: The size of the region beginning at the base address in which all pages have identical attributes in bytes
- DWORD State: The state of the pages in the region. States include MEM_COMMIT, MEM_FREE and MEM_RESERVE.
- DWORD Protect: The access protection of the pages in the region
- DWORD Type: The type of pages in the region which include MEM_IMAGE, MEM_MAPPED and MEM_PRIVATE.

VirtualAlloc

The **VirtualAlloc** function reserves, commits, or changes the state of a region of pages in the virtual address space of the calling process. Memory allocated by this function is automatically initialized to zero. If the function succeeds, the return value is the base address of the allocated region of pages.

Each page has an associated page state(Free, Reserved or Committed). The **VirtualAlloc** function can perform the following operations:

- Commit a region of reserved pages
- Reserve a region of free pages
- Simultaneously reserve and commit a region of free pages

It takes 4 arguments.

- LPVOID LpAddress: the starting address of the region to allocate.
- SIZE_T dwSize: the size of the region in bytes
- DWORD flAllocationType: The type of memory allocation. This parameter must contain one of the following values: MEM_COMMIT, MEM_RESERVE, MEM_RESET, MEM_RESET_UNDO
- DWORD flProtect: the memory protection for the region of pages to be allocated.

Example of how to use this function: [Reserving and Committing Memory - Win32 apps | Microsoft Learn](#)

GetPerformanceInfo

The **GetPerformanceInfo** function reserves the performance values contained in the **PERFORMANCE_INFORMATION** structure. If the function succeeds, the return value is TRUE. If the function fails the return value is FALSE. It takes the following argument:

- PPERFORMANCE_INFORMATION pPerformanceInformation: A pointer to a **PERFORMANCE_INFORMATION** structure that receives the performance information.
- DWORD cb: The size of the **PERFORMANCE_INFORMATION** structure, in bytes.

The **PERFORMANCE_INFORMATION** structure contains performance information. Its members include:

- DWORD cb: the size of this structure in bytes
- SIZE_T CommitTotal: the number of pages currently committed by the system.
- SIZE_T CommitLimit: The current maximum number of pages that can be committed by the system without extending the paging file(s).
- SIZE_T CommitPeak; The maximum number of pages that were simultaneously in the committed state since the last system reboot.
- SIZE_T PhysicalTotal: The amount of actual physical memory, in pages.
- SIZE_T PhysicalAvailable: The amount of physical memory currently available, in pages.
- SIZE_T SystemCache: The amount of system cache memory, in pages. This is the size of the standby list plus the system working set.
- SIZE_T KernelTotal: The sum of the memory currently in the paged and nonpaged kernel pools, in pages.
- SIZE_T KernelPaged: The memory currently in the paged kernel pool, in pages.
- SIZE_T KernelNonpaged: The memory currently in the nonpaged kernel pool, in pages.
- SIZE_T PageSize: The size of a page, in bytes.

- **DWORD HandleCount:**
- The current number of open handles.
- **DWORD ProcessCount:** The current number of processes.
- **DWORD ThreadCount:** The current number of threads.

GlobalMemoryStatusEx

The **GlobalMemoryStatusEx** function retrieves information about the systems current usage of both physical and virtual memory. If the function succeeds the return value is nonzero. It accepts one argument:

- **LPMEMORYSTATUSEX lpBuffer:** A pointer to a **MEMORYSTATUSEX** structure that receives information about memory availability.

The **MEMORYSTATUSEX** structure contains information about the current state of both physical and virtual memory, including extended memory. Its members include:

- **DWORD dwLength:** The size of the structure in bytes. This member must be set before calling **GlobalMemoryStatusEx**
- **DWORD dwMemoryLoad:** A number between 0 and 100 that specifies the approximate percentage of physical memory that is in use (0 indicates no memory use and 100 indicates full memory use).
- **DWORDLONG ullTotalPhys:** The amount of actual physical memory in bytes.
- **DWORDLONG ullAvailPhys:** The amount of physical memory currently available, in bytes. It is the sum of the size of the standby, free, and zero lists.
- **DWORDLONG ullTotalPageFile:** The current committed memory limit for the system or the current process, whichever is smaller, in bytes. To get the system-wide committed memory limit, call **GetPerformanceInfo**
- **DWORDLONG ullAvailPageFile:** The maximum amount of memory the current process can commit, in bytes. This value is equal to or smaller than the system-wide available commit value. To calculate the system-wide available commit value, call **GetPerformanceInfo** and subtract the value of **CommitTotal** from the value of **CommitLimit**.
- **DWORDLONG ullTotalVirtual:** The size of the user-mode portion of the virtual address space of the calling process, in bytes
- **DWORDLONG ullAvailVirtual:** The amount of unreserved and uncommitted memory currently in the user-mode portion of the virtual address space of the calling process, in bytes.
- **DWORDLONG ullAvailExtendedVirtual:** Reserved. This value is always 0.

```

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

// Use to convert bytes to KB
#define DIV 1024

// Specify the width of the field in which to print the numbers.
// The asterisk in the format specifier "%*I64d" takes an integer
// argument and uses it to pad and right justify the number.
#define WIDTH 7

void PrintSystemInfo()
{
    printf("\nSYSTEM INFORMATION");
    printf("\n-----\n");

    SYSTEM_INFO siSysInfo;

    // Copy the hardware information to the SYSTEM_INFO structure.
    GetSystemInfo(&siSysInfo);

    // Display the contents of the SYSTEM_INFO structure.

    printf("  OEM ID: %u\n", siSysInfo.dwOemId);
    printf("  Number of processors: %u\n",
        siSysInfo.dwNumberOfProcessors);
    printf("  Page size: %u\n", siSysInfo.dwPageSize);
    printf("  Processor type: %u\n", siSysInfo.dwProcessorType);
    printf("  Minimum application address: %lx\n",
        siSysInfo.lpMinimumApplicationAddress);
    printf("  Maximum application address: %lx\n",
        siSysInfo.lpMaximumApplicationAddress);
    printf("  Active processor mask: %u\n",
        siSysInfo.dwActiveProcessorMask);
}

void PrintMemoryInfo()
{
    printf("\nMEMORY INFORMATION\n");
    printf("-----\n");

    MEMORYSTATUSEX ms = {sizeof(MEMORYSTATUSEX)};

    ms.dwLength = sizeof(ms);

    // Retrieves information about the system's current usage of
    physical memory

```



```

GlobalMemoryStatusEx(&ms);

printf("Total memory in use: %ld%%\n", ms.dwMemoryLoad);

printf("\nTotal Physical Memory      : %8.2I64fMB \nAvailable
Physical Memory : %8.2I64fMB \nUsed Physical Memory      :
%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("Total Virtual Memory        : %8.2I64fMB \nAvailable
Virtual Memory : %8.2I64fMB \nUsed Virtual Memory        :
%8.2I64fMB \n\n", ms.ullTotalVirtual / (1024 * 1024.0),
ms.ullAvailVirtual / (1024 * 1024.0), ms.ullTotalVirtual / (1024 *
1024.0) - ms.ullAvailVirtual / (1024 * 1024.0));
}

void PrintPerformanceInfo()
{
    printf("\nPERFORMANCE INFORMATION\n");
    printf("-----\n");

    PERFORMANCE_INFORMATION siPerfInfo;

    // Copy the hardware information to the SYSTEM_INFO structure.
    GetPerformanceInfo(&siPerfInfo, siPerfInfo.cb);

    printf("Commit Total\t\t: %d pages\n", siPerfInfo.CommitTotal);
    printf("Commit Limit\t\t: %d pages\n", siPerfInfo.CommitLimit);
    printf("Commit Peak\t\t: %d pages\n", siPerfInfo.CommitPeak);
    printf("Physical Total\t\t: %d pages\n",
siPerfInfo.PhysicalTotal);
    printf("Physical Available\t: %d pages\n",
siPerfInfo.PhysicalAvailable);
    printf("System Cache\t\t: %d pages\n", siPerfInfo.SystemCache);
    printf("Kernel Total\t\t: %d pages\n", siPerfInfo.KernelTotal);
    printf("Kernel Paged\t\t: %d pages\n", siPerfInfo.KernelPaged);
    printf("Kernel Nonpaged\t\t: %d pages\n",
siPerfInfo.KernelNonpaged);
    printf("Page Size\t\t: %d MB\n", siPerfInfo.PageSize / (1024 *
1024.0));
    printf("Handle Count\t\t: %d handles\n",
siPerfInfo.HandleCount);
    printf("Process Count\t\t: %d processes\n",
siPerfInfo.ProcessCount);
    printf("Thread Count\t\t: %d threads\n",
siPerfInfo.ThreadCount);
}

void PrintRunningProcesses( void )
{
    printf("\nCURRENTLY RUNNING PROCESSES\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("LAB 4: MEMORY MONITORING\n");

    PrintSystemInfo();
    PrintMemoryInfo();
    PrintPerformanceInfo();

    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));
    if (argc != 3) /* argc should be 2 for correct execution */
    {
        printf("Usage in memorymonitoring: %s [cmdline]\n",
argv[0]);
        return;
    }
    // Start the child process.
    if (!CreateProcess(NULL,      // No module name (use command
line)
                        argv[1], // Command line
                        NULL,     // Process handle not inheritable
                        NULL,     // Thread handle not inheritable
                        FALSE,    // Set handle inheritance to FALSE
                        0,        // No creation flags
                        NULL,     // Use parent's environment block
                        NULL,     // Use parent's starting directory
                        &si,      // Pointer to STARTUPINFO structure
                        &pi)      // Pointer to PROCESS_INFORMATION
        structure
    )
    {
        printf("CreateProcess failed (%d).\n", GetLastError());
        return;
    }

    PrintRunningProcesses();

    // Just ask tasklist command

```

```
Sleep(5000);  
return 0;  
}
```

Results and Analysis [Windows]

```
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: 7ffeffff  
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>
```

Execution [Linux]

The top (table of processes) command shows a real-time view of running process in Linux. By default it sorts the process list by the %CPU column. The following commands can be used to sort using a different column:

- P. sort by the %CPU column
- T. sort by the TIME+ column
- M. sort by the %MEM column

top SORTED BY %CPU USING P SUBCOMMAND

```
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+ |


```

top SORTED BY TIME+ USING T SUBCOMMAND

```
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+

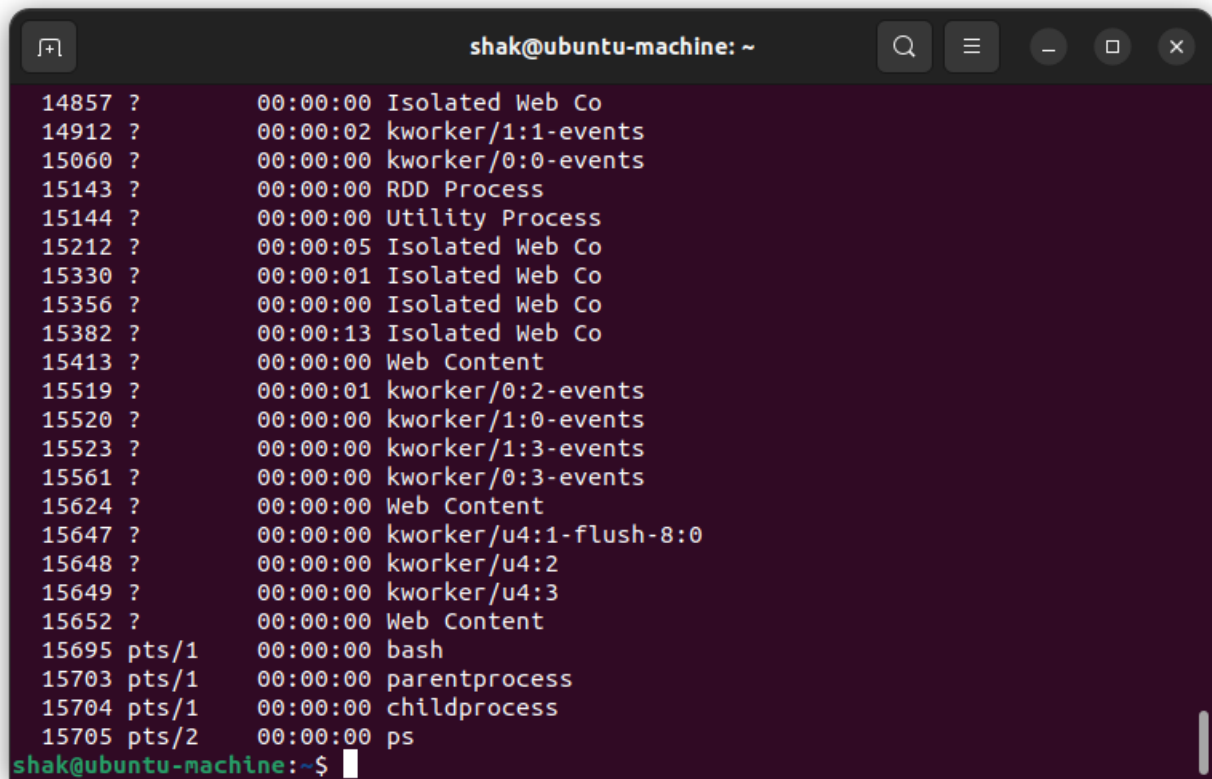
top SORTED BY %MEM USING M SUBCOMMAND

```
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+

USING ps -A TO FIND THE PID OF PARENTPROCESS

The *ps* command allows you to list the status of processes running on your system easily.

A terminal window titled 'shak@ubuntu-machine: ~' with standard window controls. It displays the output of the 'ps -A' command, listing system processes. The output is a table with columns for PID, PPID, and process name. The processes listed include 'Isolated Web Co', 'kworker', 'RDD Process', 'Utility Process', 'Web Content', 'bash', 'parentprocess', 'childprocess', and 'ps'. The 'parentprocess' entry has a PID of 15703 and a PPID of pts/1. The 'childprocess' entry has a PID of 15704 and a PPID of pts/1. The 'ps' entry has a PID of 15705 and a PPID of pts/2. The terminal prompt is 'shak@ubuntu-machine:~\$' with a cursor.

```
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:~$
```

pid of parentprocess – 15703. which is confirmed in the output of the parentprocess command


```
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 Xiaqila
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$
```

USING top -p pid TO CHECK THE STATUS OF PARENTPROCESS

```
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:~$
```

Use pmap -d pid to view the memory usage of ProcessParent

The `pmap` command in Linux is used to display the memory map of a process.

```
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-- 0000000000001000 008:00008 parentprocess
000055926e339000      4 r---- 0000000000002000 008:00008 parentprocess
000055926e33a000      4 r---- 0000000000002000 008:00008 parentprocess
000055926e33b000      4 rw--- 0000000000003000 008:00008 parentprocess
000055926e897000     132 rw--- 0000000000000000 000:00000 [ anon ]
00007fb6d1000000     160 r---- 0000000000000000 008:00009 libc.so.6
00007fb6d1028000    1620 r-x-- 0000000000028000 008:00009 libc.so.6
00007fb6d11bd000     352 r---- 000000000001bd000 008:00009 libc.so.6
00007fb6d1215000      16 r---- 00000000000214000 008:00009 libc.so.6
00007fb6d1219000       8 rw--- 00000000000218000 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 ]
fffffffff600000      4 --x-- 0000000000000000 000:00000 [ anon ]
mapped: 2776K    writeable/private: 356K    shared: 0K
shak@ubuntu-machine:~$
```

Reference:

- <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#
- [How Random Access Memory \(RAM\) affects performance | Dell US](#)
- [Physical and Virtual Memory in Windows 10 - Microsoft Community](#)
- [Page State - Win32 apps | Microsoft Learn](#)
- [MEMORY_BASIC_INFORMATION \(winnt.h\) - Win32 apps | Microsoft Learn](#)