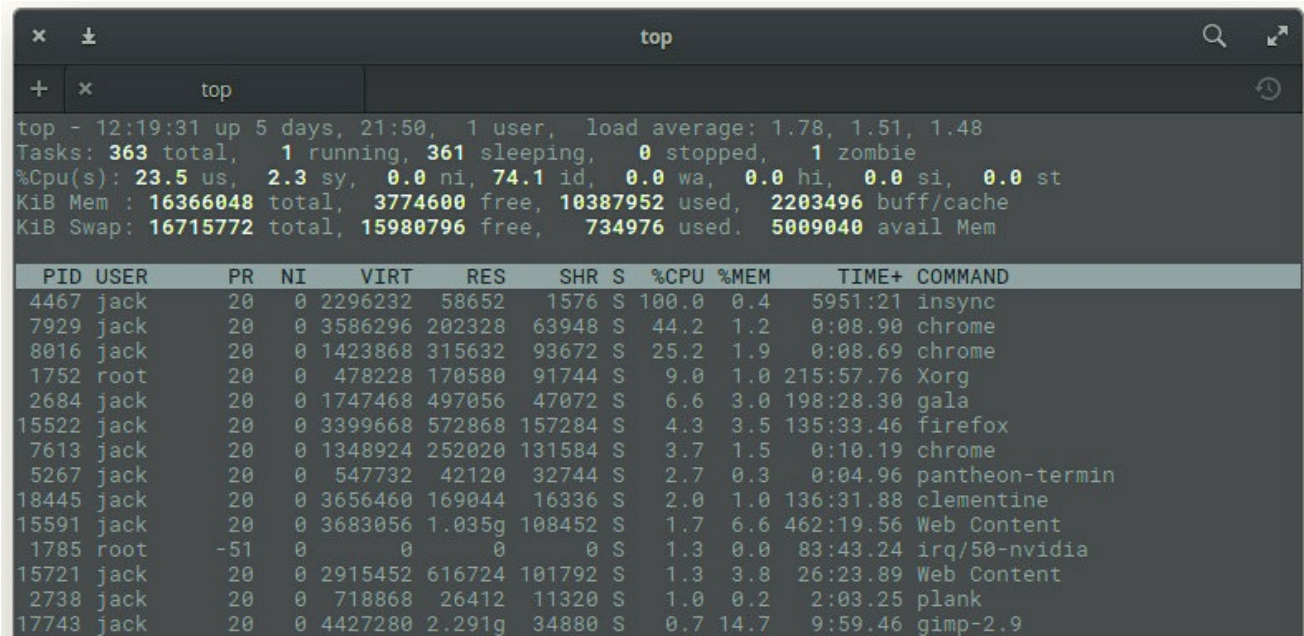


5 Commands for Checking Memory Usage in Linux

 linux.com/tutorials/5-commands-checking-memory-usage-linux/

June 15,
2018



```
top - 12:19:31 up 5 days, 21:50, 1 user, load average: 1.78, 1.51, 1.48
Tasks: 363 total, 1 running, 361 sleeping, 0 stopped, 1 zombie
%Cpu(s): 23.5 us, 2.3 sy, 0.0 ni, 74.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 16366048 total, 3774600 free, 10387952 used, 2203496 buff/cache
KiB Swap: 16715772 total, 15980796 free, 734976 used, 5009040 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 4467 jack      20   0 2296232   58652   1576 S   100.0   0.4   5951:21 insync
 7929 jack      20   0 3586296 202328   63948 S    44.2   1.2    0:08.90 chrome
 8016 jack      20   0 1423868 315632   93672 S    25.2   1.9    0:08.69 chrome
1752 root        20   0  478228 170580   91744 S     9.0   1.0   215:57.76 Xorg
 2684 jack      20   0 1747468 497056   47072 S     6.6   3.0   198:28.30 gala
15522 jack      20   0 3399668 572868  157284 S     4.3   3.5   135:33.46 firefox
 7613 jack      20   0 1348924 252020  131584 S     3.7   1.5    0:10.19 chrome
 5267 jack      20   0  547732  42120   32744 S     2.7   0.3    0:04.96 pantheon-termin
18445 jack      20   0 3656460 169044   16336 S     2.0   1.0   136:31.88 clementine
15591 jack      20   0 3683056 1.035g  108452 S     1.7   6.6   462:19.56 Web Content
 1785 root      -51   0         0         0         0 S     1.3   0.0    83:43.24 irq/50-nvidia
15721 jack      20   0 2915452 616724  101792 S     1.3   3.8    26:23.89 Web Content
 2738 jack      20   0  718868  26412   11320 S     1.0   0.2    2:03.25 plank
17743 jack      20   0 4427280 2.291g   34880 S     0.7  14.7    9:59.46 gimp-2.9
```

The Linux operating system includes a plethora of tools, all of which are ready to help you administer your systems. From simple file and directory tools to very complex security commands, there's not much you can't do on Linux. And, although regular desktop users may not need to become familiar with these tools at the command line, they're mandatory for Linux admins. Why? First, you will have to work with a GUI-less Linux server at some point. Second, command-line tools often offer far more power and flexibility than their GUI alternative.

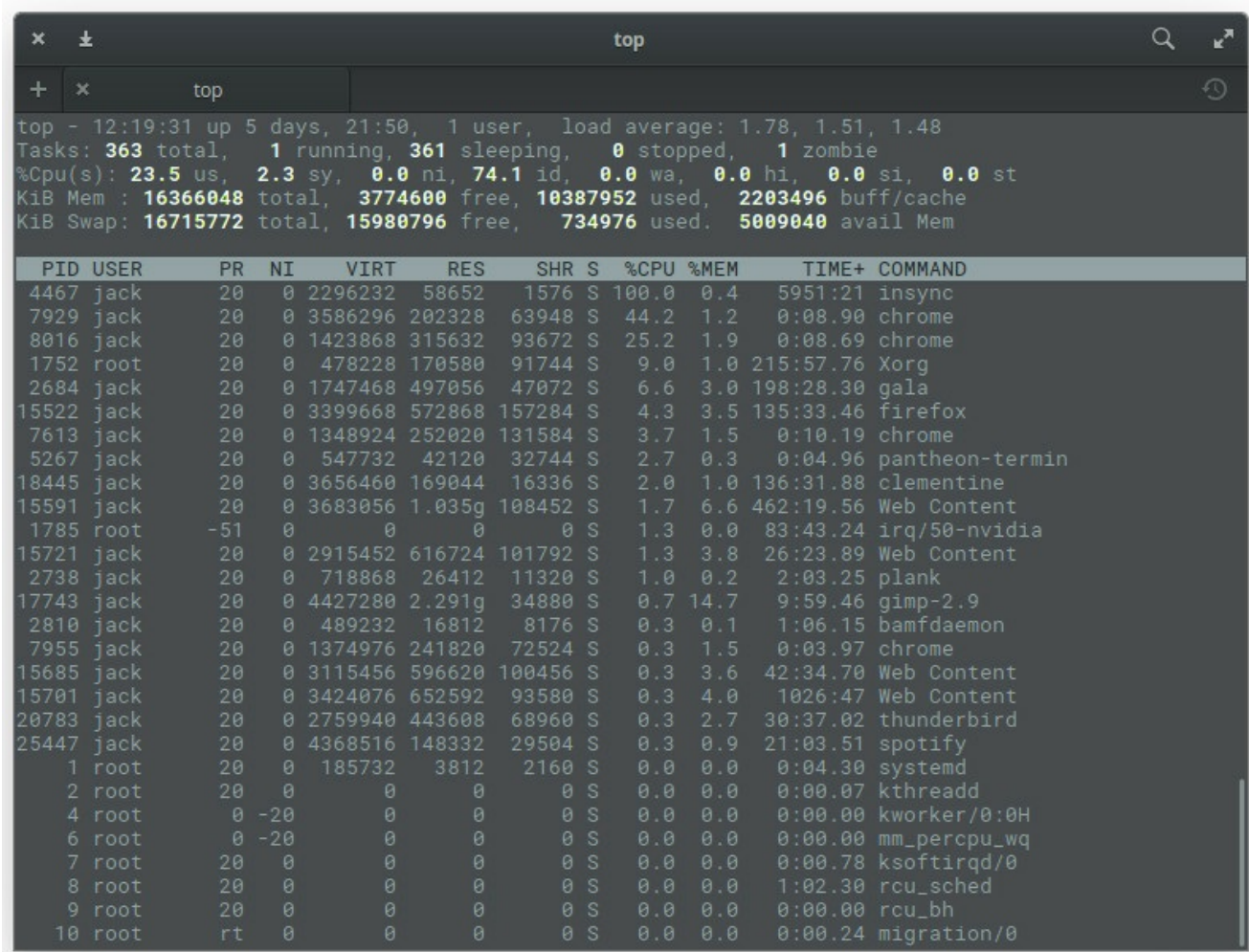
Determining memory usage is a skill you might need should a particular app go rogue and commandeer system memory. When that happens, it's handy to know you have a variety of tools available to help you troubleshoot. Or, maybe you need to gather information about a Linux swap partition or detailed information about your installed RAM? There are commands for that as well. Let's dig into the various Linux command-line tools to help you check into system memory usage. These tools aren't terribly hard to use, and in this article, I'll show you five different ways to approach the problem.

I'll be demonstrating on the [Ubuntu Server 18.04 platform](#). You should, however, find all of these commands available on your distribution of choice. Even better, you shouldn't need to install a single thing (as most of these tools are included).

With that said, let's get to work.

top

I want to start out with the most obvious tool. The `top` command provides a dynamic, real-time view of a running system. Included in that system summary is the ability to check memory usage on a per-process basis. That's very important, as you could easily have multiple iterations of the same command consuming different amounts of memory. Although you won't find this on a headless server, say you've opened Chrome and noticed your system slowing down. Issue the `top` command to see that Chrome has numerous processes running (one per tab – Figure 1).



```
top - 12:19:31 up 5 days, 21:50, 1 user, load average: 1.78, 1.51, 1.48
Tasks: 363 total, 1 running, 361 sleeping, 0 stopped, 1 zombie
%Cpu(s): 23.5 us, 2.3 sy, 0.0 ni, 74.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 16366048 total, 3774600 free, 10387952 used, 2203496 buff/cache
KiB Swap: 16715772 total, 15980796 free, 734976 used, 5009040 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4467	jack	20	0	2296232	58652	1576	S	100.0	0.4	5951:21	insync
7929	jack	20	0	3586296	202328	63948	S	44.2	1.2	0:08.90	chrome
8016	jack	20	0	1423868	315632	93672	S	25.2	1.9	0:08.69	chrome
1752	root	20	0	478228	170580	91744	S	9.0	1.0	215:57.76	Xorg
2684	jack	20	0	1747468	497056	47072	S	6.6	3.0	198:28.30	gala
15522	jack	20	0	3399668	572868	157284	S	4.3	3.5	135:33.46	firefox
7613	jack	20	0	1348924	252020	131584	S	3.7	1.5	0:10.19	chrome
5267	jack	20	0	547732	42120	32744	S	2.7	0.3	0:04.96	pantheon-termin
18445	jack	20	0	3656460	169044	16336	S	2.0	1.0	136:31.88	clementine
15591	jack	20	0	3683056	1.035g	108452	S	1.7	6.6	462:19.56	Web Content
1785	root	-51	0	0	0	0	S	1.3	0.0	83:43.24	irq/50-nvidia
15721	jack	20	0	2915452	616724	101792	S	1.3	3.8	26:23.89	Web Content
2738	jack	20	0	718868	26412	11320	S	1.0	0.2	2:03.25	plank
17743	jack	20	0	4427280	2.291g	34880	S	0.7	14.7	9:59.46	gimp-2.9
2810	jack	20	0	489232	16812	8176	S	0.3	0.1	1:06.15	bamfdaemon
7955	jack	20	0	1374976	241820	72524	S	0.3	1.5	0:03.97	chrome
15685	jack	20	0	3115456	596620	100456	S	0.3	3.6	42:34.70	Web Content
15701	jack	20	0	3424076	652592	93580	S	0.3	4.0	1026:47	Web Content
20783	jack	20	0	2759940	443608	68960	S	0.3	2.7	30:37.02	thunderbird
25447	jack	20	0	4368516	148332	29504	S	0.3	0.9	21:03.51	spotify
1	root	20	0	185732	3812	2160	S	0.0	0.0	0:04.30	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.07	kthreadd
4	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
6	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	mm_percpu_wq
7	root	20	0	0	0	0	S	0.0	0.0	0:00.78	ksoftirqd/0
8	root	20	0	0	0	0	S	0.0	0.0	1:02.30	rcu_sched
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.24	migration/0

Figure 1: Multiple instances of Chrome appearing in the `top` command.

Chrome isn't the only app to show multiple processes. You see the Firefox entry in Figure 1? That's the primary process for Firefox, whereas the Web Content processes are the open tabs. At the top of the output, you'll see the system statistics. On my machine (a [System76 Leopard Extreme](#)), I have a total of 16GB of RAM available, of which just over 10GB is in use. You can then comb through the list and see what percentage of memory each process is using.

One of the things *top* is very good for is discovering Process ID (PID) numbers of services that might have gotten out of hand. With those PIDs, you can then set about to troubleshoot (or kill) the offending tasks.

If you want to make *top* a bit more memory-friendly, issue the command *top -o %MEM*, which will cause *top* to sort all processes by memory used (Figure 2).

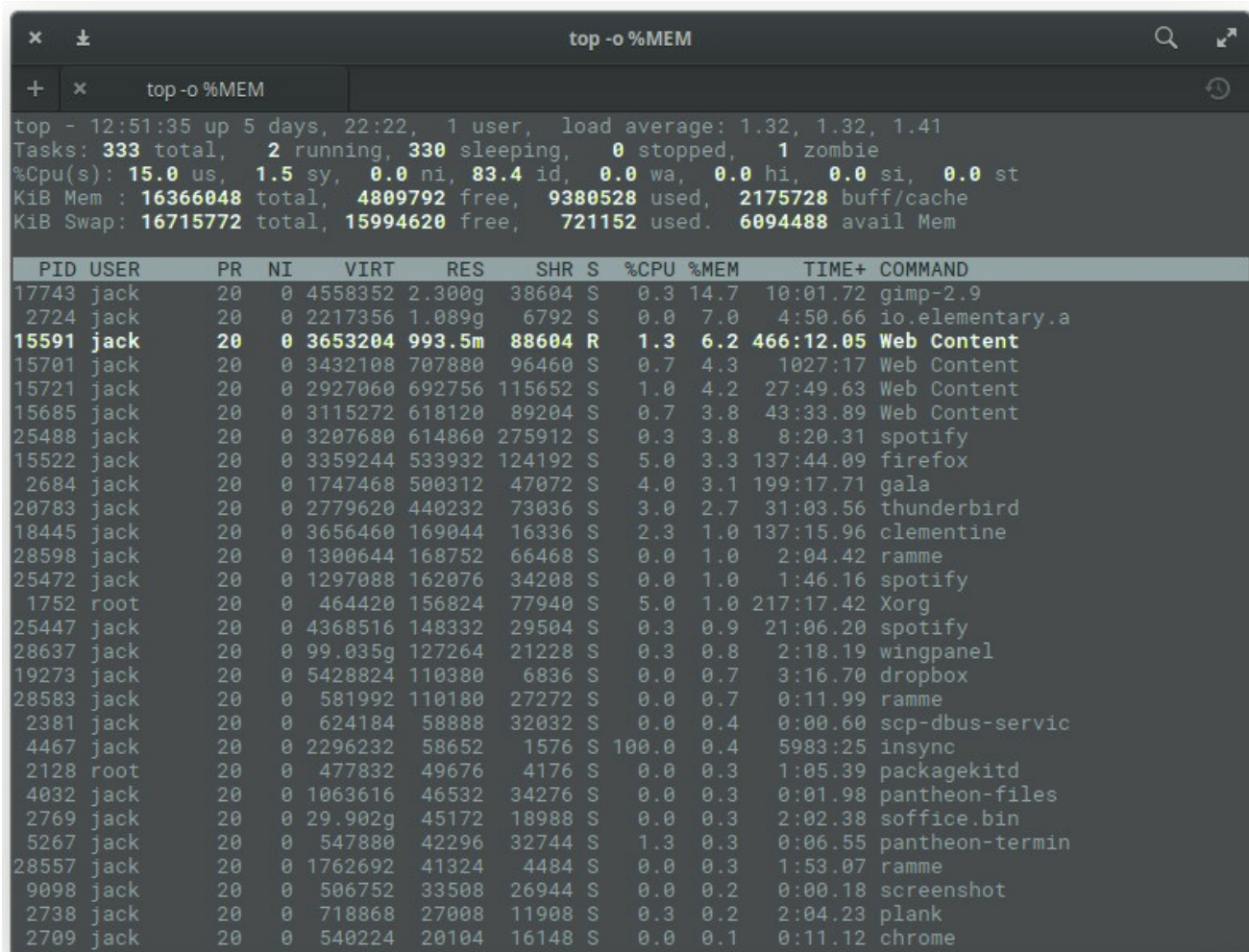
The image shows a terminal window titled 'top -o %MEM'. It displays the output of the 'top' command sorted by memory usage. The header row is: PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND. The first few rows are: 17743 jack 20 0 4558352 2.300g 38604 S 0.3 14.7 10:01.72 gimp-2.9; 2724 jack 20 0 2217356 1.089g 6792 S 0.0 7.0 4:50.66 io.elementary.a; 15591 jack 20 0 3653204 993.5m 88604 R 1.3 6.2 466:12.05 Web Content; 15701 jack 20 0 3432108 707880 96460 S 0.7 4.3 10:27.17 Web Content; 15721 jack 20 0 2927060 692756 115652 S 1.0 4.2 27:49.63 Web Content; 15685 jack 20 0 3115272 618120 89204 S 0.7 3.8 43:33.89 Web Content; 25488 jack 20 0 3207680 614860 275912 S 0.3 3.8 8:20.31 spotify; 15522 jack 20 0 3359244 533932 124192 S 5.0 3.3 137:44.09 firefox; 2684 jack 20 0 1747468 500312 47072 S 4.0 3.1 199:17.71 gala; 20783 jack 20 0 2779620 440232 73036 S 3.0 2.7 31:03.56 thunderbird; 18445 jack 20 0 3656460 169044 16336 S 2.3 1.0 137:15.96 clementine; 28598 jack 20 0 1300644 168752 66468 S 0.0 1.0 2:04.42 ramme; 25472 jack 20 0 1297088 162076 34208 S 0.0 1.0 1:46.16 spotify; 1752 root 20 0 464420 156824 77940 S 5.0 1.0 217:17.42 Xorg; 25447 jack 20 0 4368516 148332 29504 S 0.3 0.9 21:06.20 spotify; 28637 jack 20 0 99.035g 127264 21228 S 0.3 0.8 2:18.19 wingpanel; 19273 jack 20 0 5428824 110380 6836 S 0.0 0.7 3:16.70 dropbox; 28583 jack 20 0 581992 110180 27272 S 0.0 0.7 0:11.99 ramme; 2381 jack 20 0 624184 58888 32032 S 0.0 0.4 0:00.60 scp-dbus-servic; 4467 jack 20 0 2296232 58652 1576 S 100.0 0.4 5983:25 insync; 2128 root 20 0 477832 49676 4176 S 0.0 0.3 1:05.39 packagekitd; 4032 jack 20 0 1063616 46532 34276 S 0.0 0.3 0:01.98 pantheon-files; 2769 jack 20 0 29.902g 45172 18988 S 0.0 0.3 2:02.38 soffice.bin; 5267 jack 20 0 547880 42296 32744 S 1.3 0.3 0:06.55 pantheon-termin; 28557 jack 20 0 1762692 41324 4484 S 0.0 0.3 1:53.07 ramme; 9098 jack 20 0 506752 33508 26944 S 0.0 0.2 0:00.18 screenshot; 2738 jack 20 0 718868 27008 11908 S 0.3 0.2 2:04.23 plank; 2709 jack 20 0 540224 20104 16148 S 0.0 0.1 0:11.12 chrome; The summary at the top shows: top - 12:51:35 up 5 days, 22:22, 1 user, load average: 1.32, 1.32, 1.41; Tasks: 333 total, 2 running, 330 sleeping, 0 stopped, 1 zombie; %Cpu(s): 15.0 us, 1.5 sy, 0.0 ni, 83.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st; KiB Mem : 16366048 total, 4809792 free, 9380528 used, 2175728 buff/cache; KiB Swap: 16715772 total, 15994620 free, 721152 used. 6094488 avail Mem

Figure 2: Sorting process by memory used in *top*.

The *top* command also gives you a real-time update on how much of your swap space is being used.

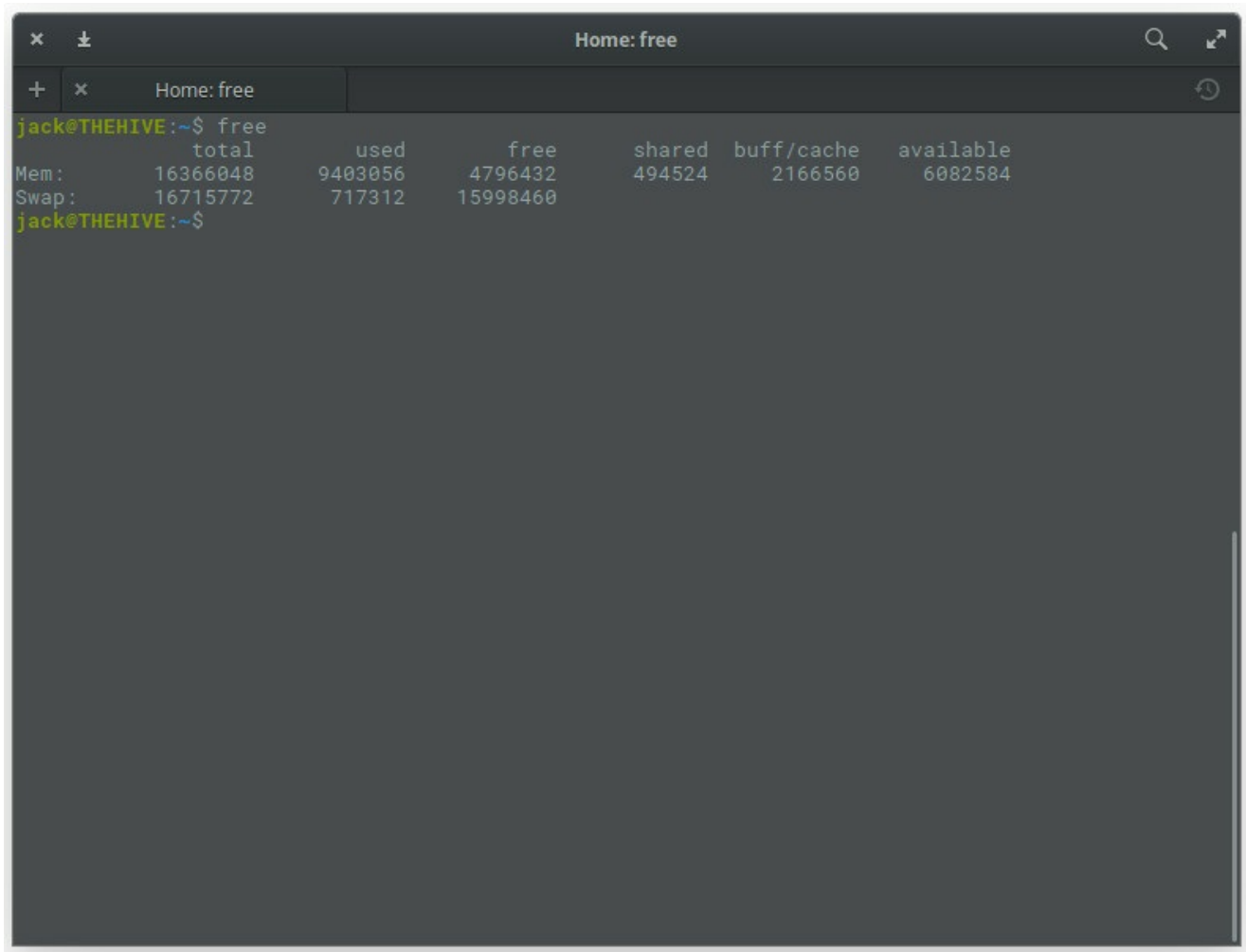
free

Sometimes, however, *top* can be a bit much for your needs. You may only need to see the amount of free and used memory on your system. For that, there is the *free* command. The *free* command displays:

- Total amount of free and used physical memory

- Total amount of swap memory in the system
- Buffers and caches used by the kernel

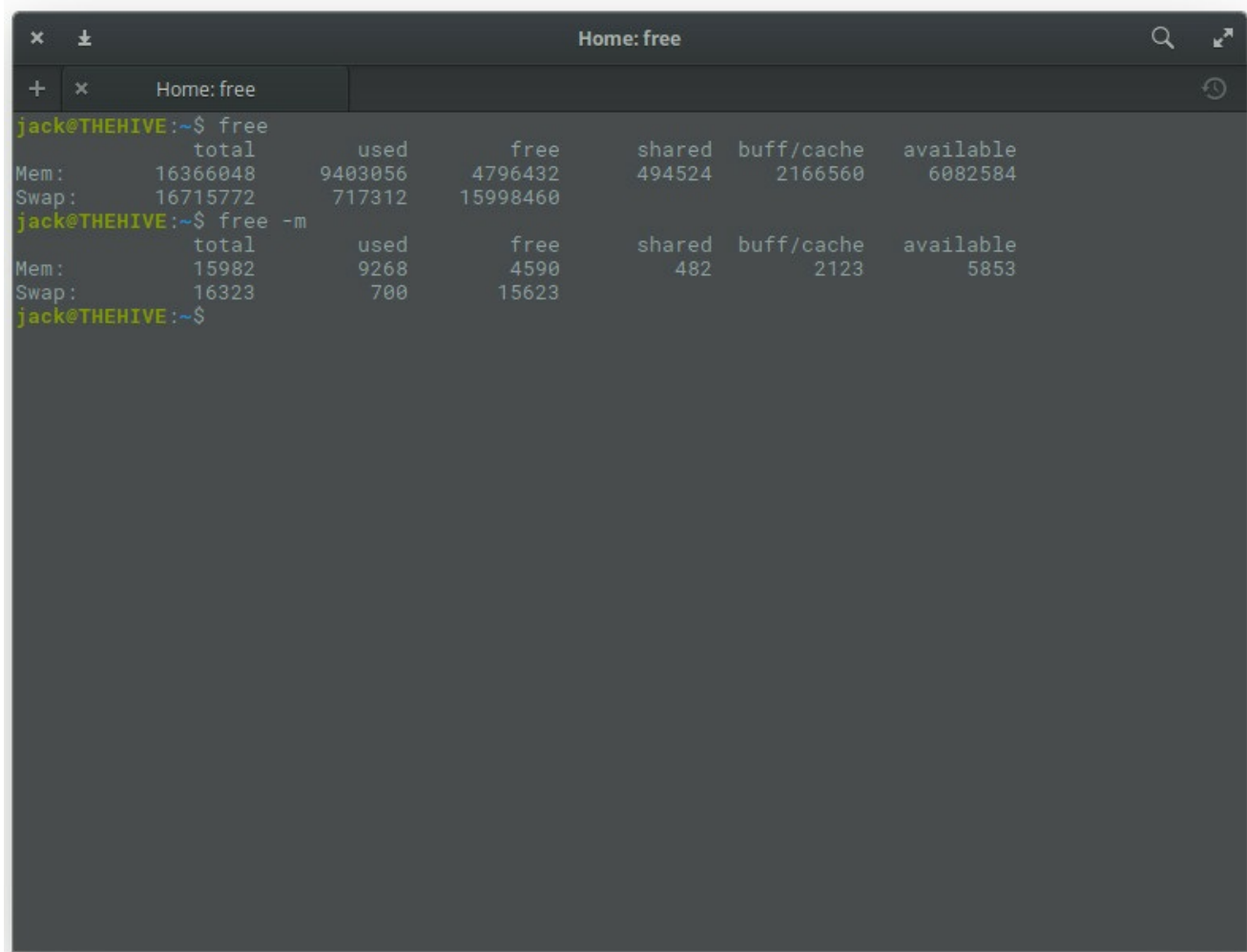
From your terminal window, issue the command *free*. The output of this command is not in real time. Instead, what you'll get is an instant snapshot of the free and used memory in that moment (Figure 3).

A terminal window titled "Home: free" showing the output of the 'free' command. The output is a table with 6 columns: total, used, free, shared, buff/cache, and available. The rows are for Mem and Swap. The prompt is jack@THEHIVE:~\$.

	total	used	free	shared	buff/cache	available
Mem:	16366048	9403056	4796432	494524	2166560	6082584
Swap:	16715772	717312	15998460			

Figure 3: The output of the free command is simple and clear.

You can, of course, make *free* a bit more user-friendly by adding the *-m* option, like so:*free -m*. This will report the memory usage in MB (Figure 4).

A terminal window titled "Home: free" showing the output of the 'free' and 'free -m' commands. The first command 'free' shows memory usage in bytes, and the second command 'free -m' shows memory usage in megabytes. The output is formatted as a table with columns: total, used, free, shared, buff/cache, and available.

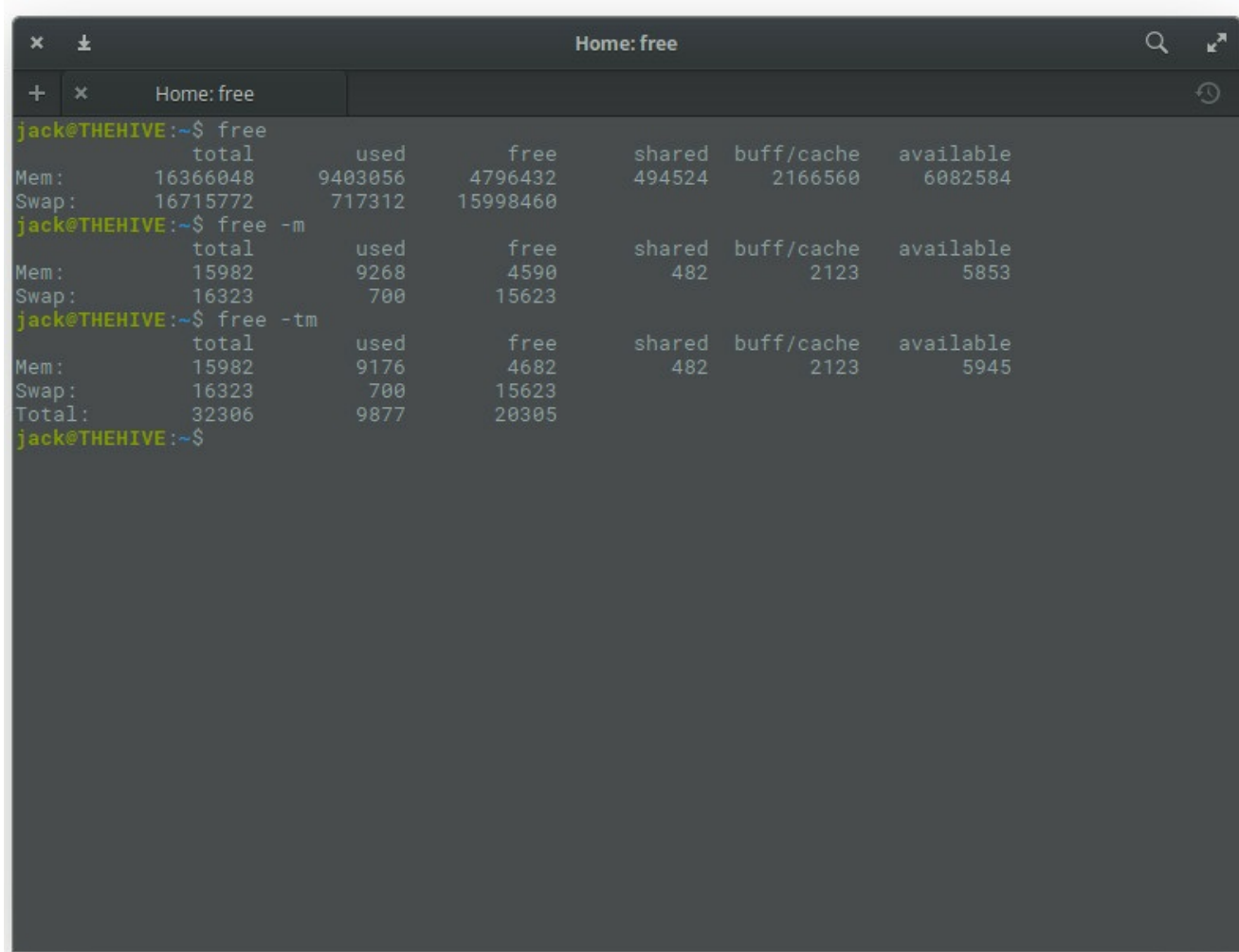
```
jack@THEHIVE:~$ free
              total        used        free      shared  buff/cache   available
Mem:      16366048     9403056     4796432      494524     2166560     6082584
Swap:     16715772      717312    15998460

jack@THEHIVE:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           15982         9268         4590         482         2123         5853
Swap:          16323          700        15623
```

Figure 4: The output of the free command in a more human-readable form.

Of course, if your system is even remotely modern, you'll want to use the `-g` option (gigabytes), as in `free -g`.

If you need memory totals, you can add the `t` option like so: `free -mt`. This will simply total the amount of memory in columns (Figure 5).



```
jack@THEHIVE:~$ free
              total        used        free      shared  buff/cache   available
Mem:      16366048     9403056     4796432      494524      2166560      6082584
Swap:     16715772     717312      15998460

jack@THEHIVE:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           15982         9268         4590         482         2123         5853
Swap:          16323          700        15623

jack@THEHIVE:~$ free -tm
              total        used        free      shared  buff/cache   available
Mem:           15982         9176         4682         482         2123         5945
Swap:          16323          700        15623
Total:         32306         9877        20305

jack@THEHIVE:~$
```

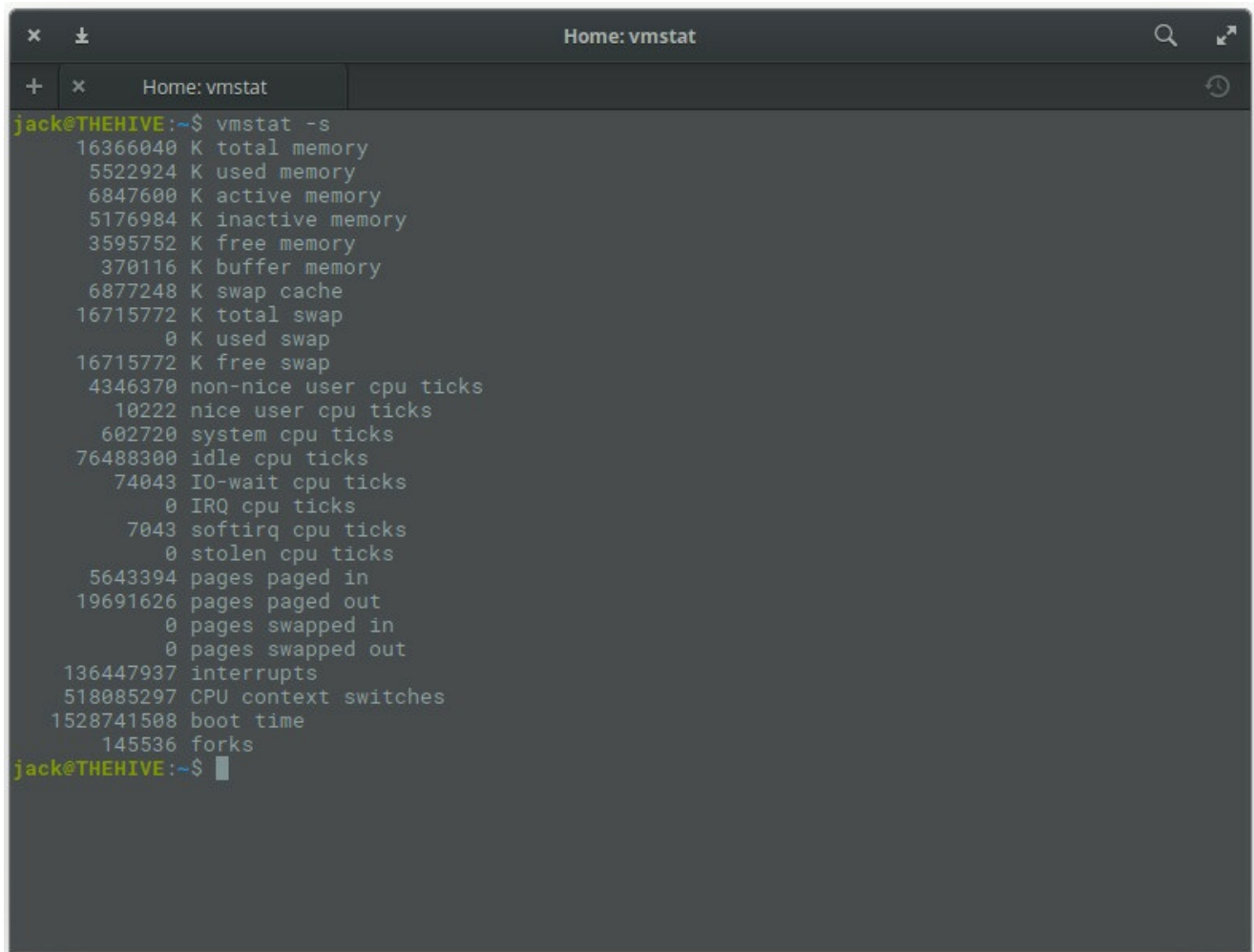
Figure 5: Having free total your memory columns for you.

vmstat

Another very handy tool to have at your disposal is *vmstat*. This particular command is a one-trick pony that reports virtual memory statistics. The *vmstat* command will report stats on:

- Processes
- Memory
- Paging
- Block IO
- Traps
- Disks
- CPU

The best way to issue *vmstat* is by using the *-s* switch, like *vmstat -s*. This will report your stats in a single column (which is so much easier to read than the default report). The *vmstat* command will give you more information than you need (Figure 6), but more is always better (in such cases).

A terminal window titled "Home: vmstat" showing the output of the `vmstat -s` command. The output lists various system statistics in a single column, including memory usage (total, used, active, inactive, free, buffer), swap usage (total, used, free), CPU ticks (non-nice user, nice user, system, idle, IO-wait, IRQ, softirq, stolen), pages paged in/out, pages swapped in/out, interrupts, CPU context switches, boot time, and forks. The prompt is `jack@THEHIVE:~$`.

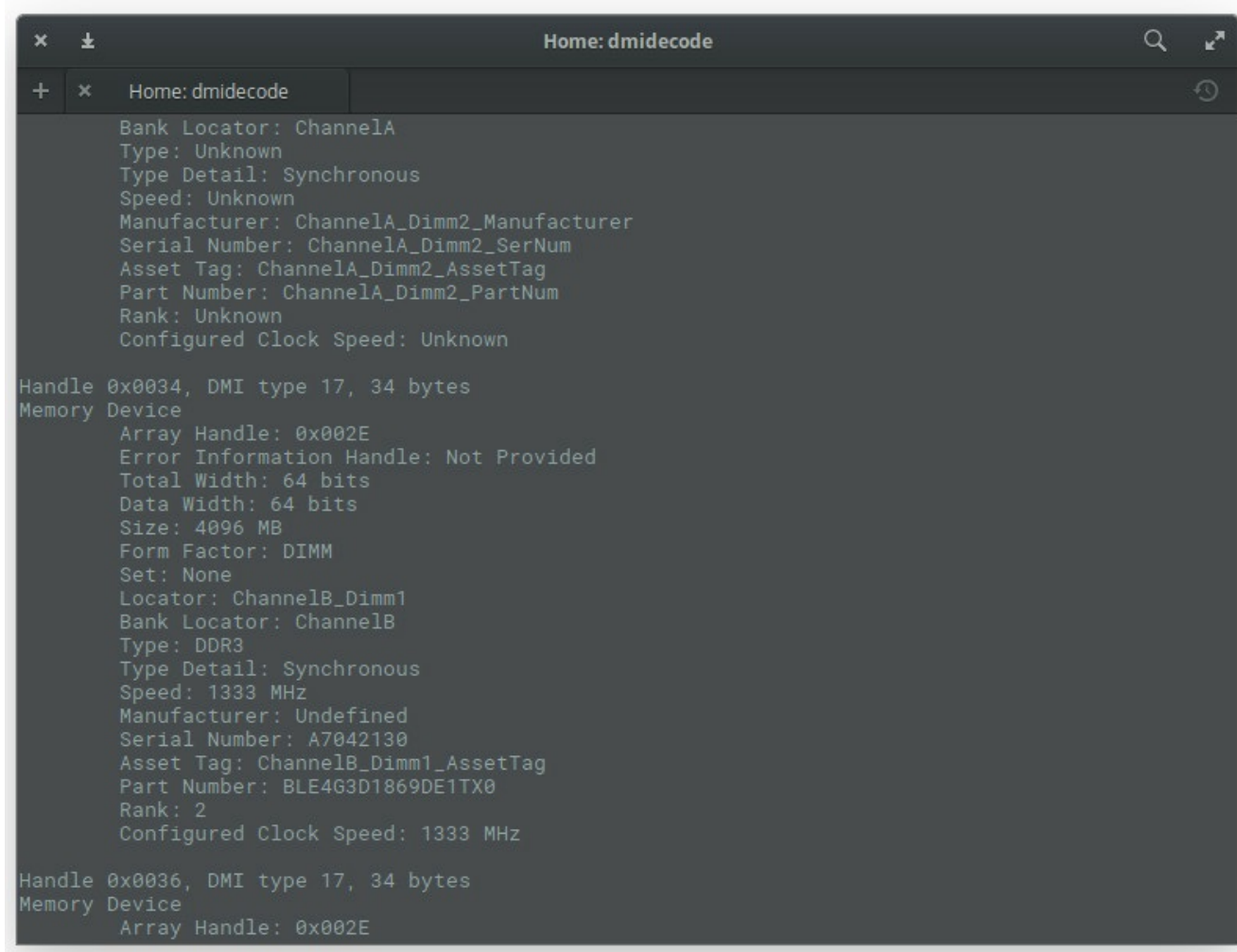
```
jack@THEHIVE:~$ vmstat -s
16366040 K total memory
5522924 K used memory
6847600 K active memory
5176984 K inactive memory
3595752 K free memory
370116 K buffer memory
6877248 K swap cache
16715772 K total swap
0 K used swap
16715772 K free swap
4346370 non-nice user cpu ticks
10222 nice user cpu ticks
602720 system cpu ticks
76488300 idle cpu ticks
74043 IO-wait cpu ticks
0 IRQ cpu ticks
7043 softirq cpu ticks
0 stolen cpu ticks
5643394 pages paged in
19691626 pages paged out
0 pages swapped in
0 pages swapped out
136447937 interrupts
518085297 CPU context switches
1528741508 boot time
145536 forks
jack@THEHIVE:~$
```

Figure 6: Using the *vmstat* command to check memory usage.

dmidecode

What if you want to find out detailed information about your installed system RAM? For that, you could use the *dmidecode* command. This particular tool is the DMI table decoder, which dumps a system's DMI table contents into a human-readable format. If you're unsure as to what the DMI table is, it's a means to describe what a system is made of (as well as possible evolutions for a system).

To run the *dmidecode* command, you do need *sudo* privileges. So issue the command *sudo dmidecode -t 17*. The output of the command (Figure 7) can be lengthy, as it displays information for all memory-type devices. So if you don't have the ability to scroll, you might want to send the output of that command to a file, like so: *sudo dmidecode -t 17 > dmi_info*, or pipe it to the *less* command, as in *sudo dmidecode | less*.

A terminal window titled "Home: dmidecode" with a search icon and window controls. It shows the output of the dmidecode command. The first section is for Channel A Dimm 2, showing fields like Bank Locator, Type, Speed, Manufacturer, Serial Number, Asset Tag, Part Number, Rank, and Configured Clock Speed. The second section is for Channel B Dimm 1, showing similar fields. The third section is for Channel A Dimm 1, showing fields like Bank Locator, Type, Speed, Manufacturer, Serial Number, Asset Tag, Part Number, Rank, and Configured Clock Speed. The fourth section is for Channel B Dimm 2, showing fields like Bank Locator, Type, Speed, Manufacturer, Serial Number, Asset Tag, Part Number, Rank, and Configured Clock Speed.

```
Bank Locator: ChannelA
Type: Unknown
Type Detail: Synchronous
Speed: Unknown
Manufacturer: ChannelA_Dimm2_Manufacturer
Serial Number: ChannelA_Dimm2_SerNum
Asset Tag: ChannelA_Dimm2_AssetTag
Part Number: ChannelA_Dimm2_PartNum
Rank: Unknown
Configured Clock Speed: Unknown

Handle 0x0034, DMI type 17, 34 bytes
Memory Device
Array Handle: 0x002E
Error Information Handle: Not Provided
Total Width: 64 bits
Data Width: 64 bits
Size: 4096 MB
Form Factor: DIMM
Set: None
Locator: ChannelB_Dimm1
Bank Locator: ChannelB
Type: DDR3
Type Detail: Synchronous
Speed: 1333 MHz
Manufacturer: Undefined
Serial Number: A7042130
Asset Tag: ChannelB_Dimm1_AssetTag
Part Number: BLE4G3D1869DE1TX0
Rank: 2
Configured Clock Speed: 1333 MHz

Handle 0x0036, DMI type 17, 34 bytes
Memory Device
Array Handle: 0x002E
```

Figure 7: The output of the dmidecode command.

/proc/meminfo

You might be asking yourself, “Where do these commands get this information from?”. In some cases, they get it from the `/proc/meminfo` file. Guess what? You can read that file directly with the command `less /proc/meminfo`. By using the `less` command, you can scroll up and down through that lengthy output to find exactly what you need (Figure 8).

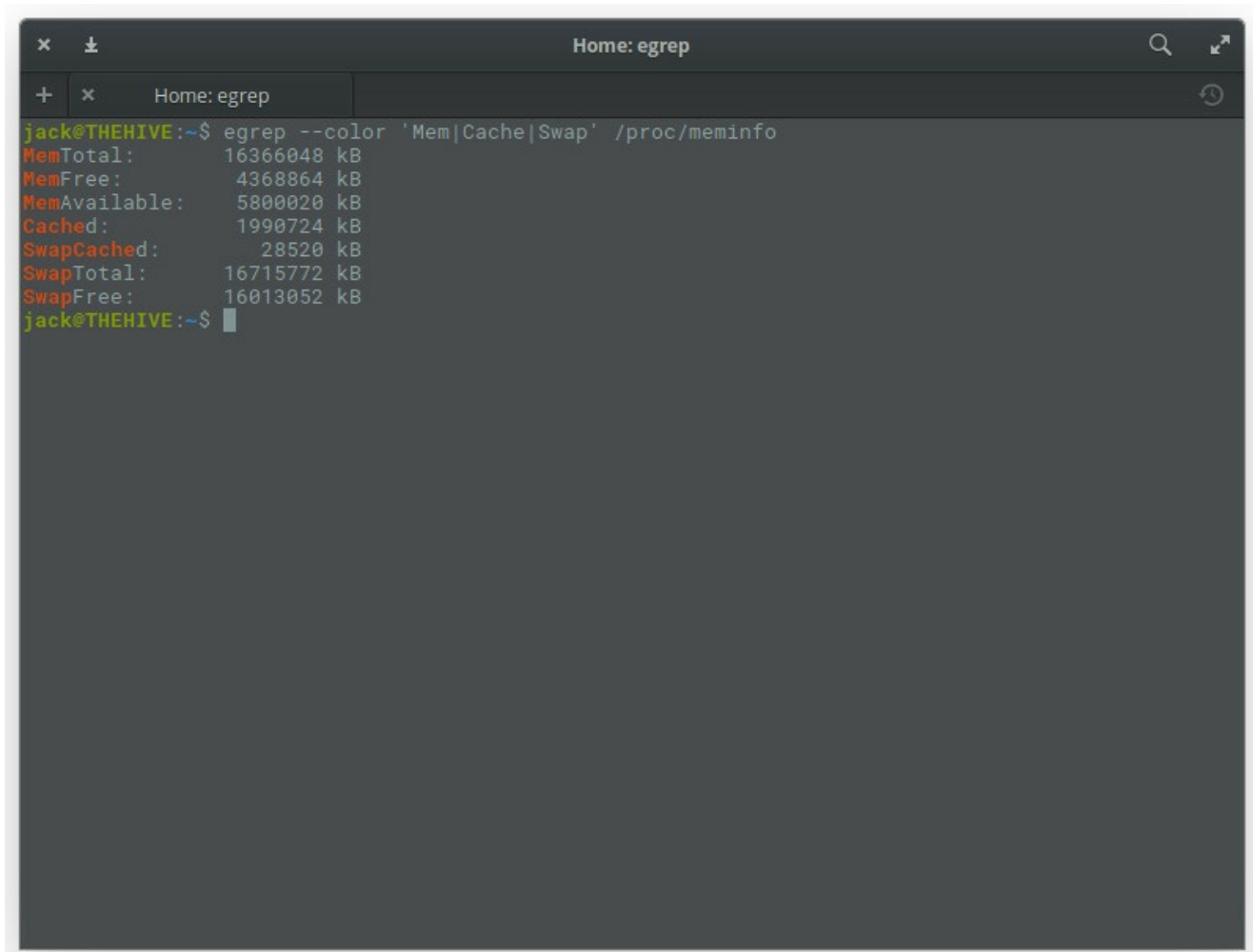

```
MemTotal:      16366048 kB
MemFree:       4616552 kB
MemAvailable:  5934828 kB
Buffers:       102796 kB
Cached:        1920072 kB
SwapCached:    23144 kB
Active:        9346904 kB
Inactive:      1814128 kB
Active(anon):  8586528 kB
Inactive(anon): 1054724 kB
Active(file):  760376 kB
Inactive(file): 759404 kB
Unevictable:   1536 kB
Mlocked:       1536 kB
SwapTotal:     16715772 kB
SwapFree:      15998716 kB
Dirty:         4 kB
Writeback:     0 kB
AnonPages:     9120848 kB
Mapped:        903316 kB
Shmem:         532960 kB
Slab:          214704 kB
SReclaimable:  104488 kB
SUnreclaim:    110216 kB
KernelStack:   18416 kB
PageTables:    87924 kB
NFS_Unstable:  0 kB
Bounce:        0 kB
WritebackTmp:  0 kB
CommitLimit:   24898796 kB
Committed_AS:  21233292 kB
VmallocTotal:  34359738367 kB
VmallocUsed:   0 kB
VmallocChunk:  0 kB
:
```

Figure 8: The output of the `less /proc/meminfo` command.

One thing you should know about */proc/meminfo*: This is not a real file. Instead */pro/meminfo* is a virtual file that contains real-time, dynamic information about the system. In particular, you'll want to check the values for:

- MemTotal
- MemFree
- MemAvailable
- Buffers
- Cached
- SwapCached
- SwapTotal
- SwapFree

If you want to get fancy with `/proc/meminfo` you can use it in conjunction with the `egrep` command like so: `egrep --color 'Mem|Cache|Swap' /proc/meminfo`. This will produce an easy to read listing of all entries that contain Mem, Cache, and Swap ... with a splash of color (Figure 9).

A terminal window titled 'Home: egrep' showing the output of the command `egrep --color 'Mem|Cache|Swap' /proc/meminfo`. The output is color-coded: 'Mem' is red, 'Cache' is orange, and 'Swap' is green. The output shows memory statistics for a system with 16GB of RAM and 16GB of swap space.

```
jack@THEHIVE:~$ egrep --color 'Mem|Cache|Swap' /proc/meminfo
MemTotal:      16366048 kB
MemFree:       4368864 kB
MemAvailable:  5800020 kB
Cached:        1990724 kB
SwapCached:    28520 kB
SwapTotal:     16715772 kB
SwapFree:      16013052 kB
jack@THEHIVE:~$
```

Figure 9: Making `/proc/meminfo` easier to read.

Keep learning

One of the first things you should do is read the manual pages for each of these commands (so *man top*, *man free*, *man vmstat*, *man dmidecode*). Starting with the man pages for commands is always a great way to learn so much more about how a tool works on Linux.

Learn more about Linux through the free ["Introduction to Linux"](#) course from The Linux Foundation and edX.