October 3, 2012

Resident and Virtual memory on Linux: A short example

Posted in Software at 06:29 by graham

Tools like `top` show processes using two kinds of memory:

- Resident memory, labelled RES: How much physical memory, how much RAM, your process is using. **RES is the important number**.
- Virtual memory, labelled VIRT: How much memory your process *thinks* it's using. Usually much bigger than RES, thanks to the Linux kernel's clever memory management.

Here's a short C program to illustrate the difference:

```
#include <stdio.h>
#include <stdlib.h>

void fill(unsigned char* addr, size_t amount) {
    unsigned long i;
    for (i = 0; i < amount; i++) {
        *(addr + i) = 42;
    }
}

int main(int argc, char **argv) {

    unsigned char *result;
    char input;
    size_t s = 1<<30;
```

Search

All posts by date
Credit card numbers
Credit card generator
SSH and OpenPGP keys
Research Papers Archive
Behaviour Change Toolkit
about

Categories

- Behaviour
- Credit card
- Finance
- Future of Web Apps
- History

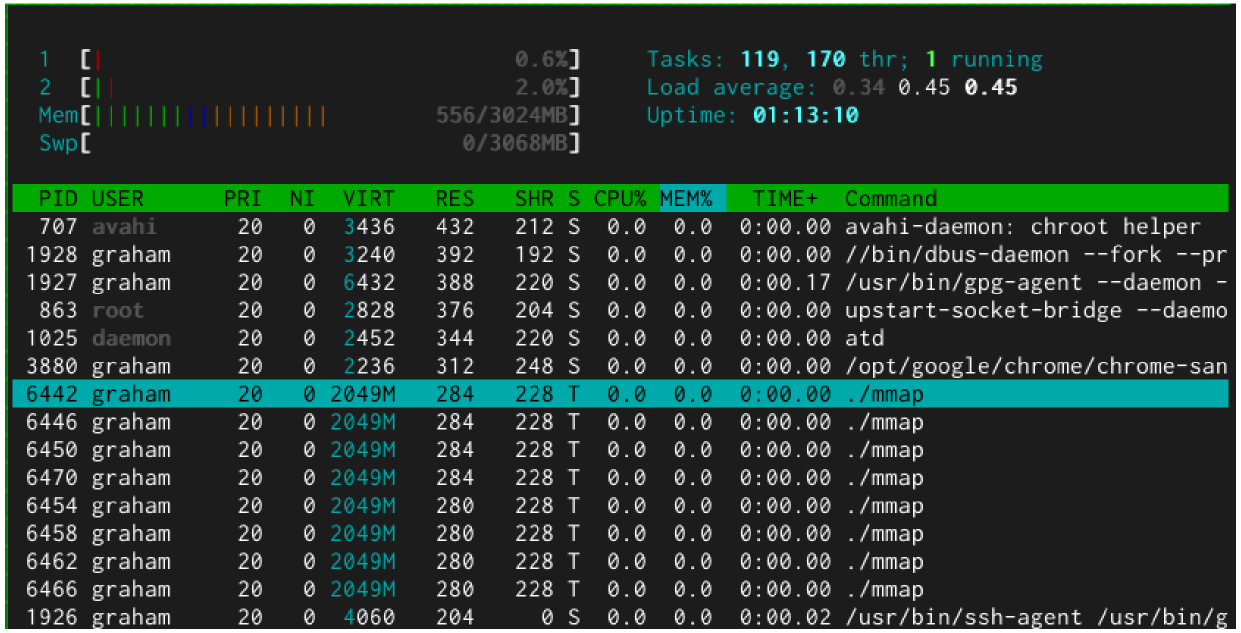
```
    result = malloc(s);
    printf("Addr: %p\n", result);
    //fill(result, s);

    scanf("%c", &input);
    return 0;
}
```

Save it as `mem.c`, compile it `cc -Wall -g mem.c -o mem` and run it.

It requests 1 Gig of memory (1 << 30), but doesn't use it. With `top` (or `htop`, which is much nicer), or even `ps -Ao rsz,vsz,cmd | grep mem` view it's memory usage. The resident size should be quite small (~280k for me), but the virtual size will be the full 1 Gig we requested.

Uncomment the `fill` line, compile and run it again. You should see the resident size get much bigger, as we're actually using the memory we requested (we're filling it with the number 42). The kernel has to give us real memory to work with.

We can request far more virtual memory than the machine can handle. Here's a screenshot where I've requested 16 Gig of memory, on a machine with only 3 Gig of RAM and 3 Gig of swap. The VIRT's sum up to way more than that machine can handle. At the top left of the screenshot you can see none of that memory has really been allocated. The kernel won't allocate it until we use it.



```
  1  [|                          0.6%]     Tasks: 119, 170 thr; 1 running
  2  [||                         2.0%]     Load average: 0.34 0.45 0.45
  Mem[||||||||||||||||||   556/3024MB]     Uptime: 01:13:10
  Swp[                       0/3068MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
  707 avahi      20   0  3436   432   212 S  0.0  0.0  0:00.00 avahi-daemon: chroot helper
 1928 graham     20   0  3240   392   192 S  0.0  0.0  0:00.00 //bin/dbus-daemon --fork --pr
 1927 graham     20   0  6432   388   220 S  0.0  0.0  0:00.17 /usr/bin/gpg-agent --daemon -
  863 root       20   0  2828   376   204 S  0.0  0.0  0:00.00 upstart-socket-bridge --daemo
 1025 daemon     20   0  2452   344   220 S  0.0  0.0  0:00.00 atd
 3880 graham     20   0  2236   312   248 S  0.0  0.0  0:00.00 /opt/google/chrome/chrome-san
 6442 graham     20   0 2049M   284   228 T  0.0  0.0  0:00.00 ./mmap
 6446 graham     20   0 2049M   284   228 T  0.0  0.0  0:00.00 ./mmap
 6450 graham     20   0 2049M   284   228 T  0.0  0.0  0:00.00 ./mmap
 6470 graham     20   0 2049M   284   228 T  0.0  0.0  0:00.00 ./mmap
 6454 graham     20   0 2049M   280   228 T  0.0  0.0  0:00.00 ./mmap
 6458 graham     20   0 2049M   280   228 T  0.0  0.0  0:00.00 ./mmap
 6462 graham     20   0 2049M   280   228 T  0.0  0.0  0:00.00 ./mmap
 6466 graham     20   0 2049M   280   228 T  0.0  0.0  0:00.00 ./mmap
 1926 graham     20   0  4060   204     0 S  0.0  0.0  0:00.02 /usr/bin/ssh-agent /usr/bin/g
```

In summary, virtual size can largely be ignored. Resident size is the important number.

9 Comments »

Michele said,
November 7, 2017 at 15:30

WOW: enlightening article in less than 2K characters!

Manu said,
May 18, 2017 at 11:02

Hi,

I didn’t understand this line “In summary, virtual size can largely be ignored. Resident size is the important number”. When I write a code which allocates and de-allocates memory , shouldn’t I be worried about the Virtual memory that I allocate and de-allocate? Since this RES memory should be taken care by the OS right.

Manu

Zhani Baramidze said,
March 16, 2016 at 09:18

Hey Graham, first of all thanks for such a nice example! my question is, I don’t understand why, after allocating 1G memory and not using it, RES doesn’t show it. After allocating doesn’t kernel knows that it is ours, even if we don’t use it? I disable gcc optimizations via -O0 but it still doesn’t show it

---

graham said,
February 9, 2016 at 20:20

@Andrew I’m using the “binary” meaning of GB, second paragraph here: https://en.wikipedia.org/wiki/Gigabyte. 1<<30 is indeed 1073741824, which is 1024*1024*1024.

Andrew F said,
February 9, 2016 at 15:07

Hey Graham, Very informative post, thanks for your contribution. I’m struggling to understand how 1<<30 = 1GB though. 1<<30=2^30=1073741824 correct? Thanks, -

Andrew

Bryan Conrad said,
September 3, 2014 at 15:49

One more giving kudos for the code example. Very informative, thanks!

Nicholas said,
April 18, 2014 at 00:35

To the point and informative–great post. Thanks!

---

christolb29 said,
February 21, 2014 at 14:13

Thanks, nice example.

Josh Enders said,
October 27, 2013 at 00:38

Hey Graham, thanks for providing the code sample. Very illustrative!

Leave a Comment

Note: Your comment will only appear on the site once I approve it manually. This can take a day or two. Thanks for taking the time to comment.

Name (required)

E-mail (required)

Submit Comment