

# CoE 135: Lab 7 Documentation (Memory)

Salmon, Paulino III I.

2015-11557

paulino.salmon@eee.upd.edu.ph

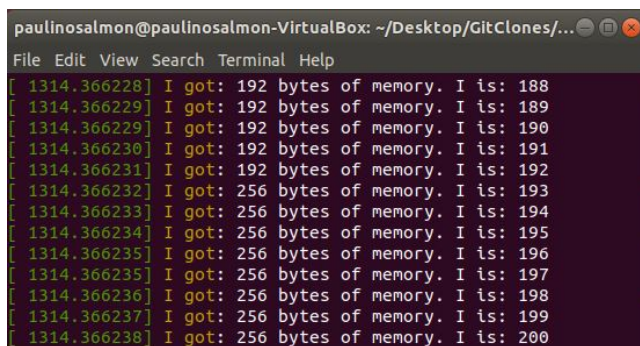
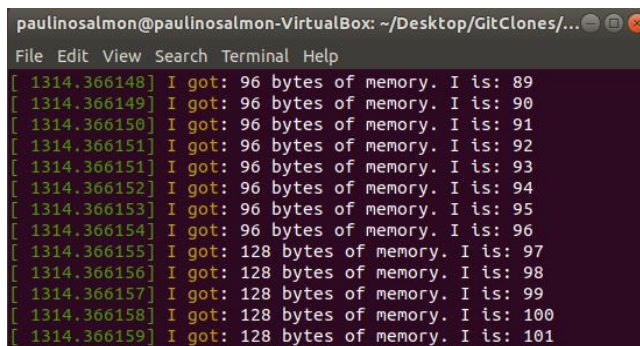
## I. PROPERTIES OF KMALLOC() (50 PTS)

- 1) Does `kmalloc()` always allocate memory sizes in powers of two?

To answer this, I created a for loop in my kernel module that would continuously just allocate and free memory used by `kmalloc()` at a maximum value of 255. This counter is to be plugged in the code line: `ptr = kmalloc(i, GFP_KERNEL);`

```
// #1
unsigned int i;
for(i = 0; i < 255; i++) {
    ptr = kmalloc(i, GFP_KERNEL);
    printk(KERN_INFO "I got: %zu
        bytes of memory. I is: %d\n",
        ksize(ptr), i);
    kfree(ptr);
}
```

Running this kernel module in `dmesg`, it shows that the allocated bytes of memory are **not always** in powers of two (See 96 bytes and 192 bytes below).



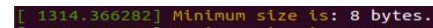
Page sizes are normally in powers of two, but the system allocates usually a little bit more than what is

asked for as according to Stack Overflow User Mike [1].

- 2) What is the minimum amount of memory `kmalloc()` can provide?

Using the code snippet below and printing the kernel module in `dmesg` afterwards:

```
// #2
printk(KERN_INFO "Minimum size is: %d
    bytes.\n", KMALLOC_MIN_SIZE);
```

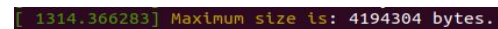


Terminal output shows that the minimum amount of memory `kmalloc()` can provide is **8 bytes**. This `KMALLOC_MIN_SIZE` constant is defined in the `slab.h` header.

- 3) What is the maximum amount of memory `kmalloc()` can provide?

Using the code snippet below and printing the kernel module in `dmesg` afterwards:

```
// #3
printk(KERN_INFO "Maximum size is: %ld
    bytes.\n", KMALLOC_MAX_SIZE);
```



Terminal output shows that the minimum amount of memory `kmalloc()` can provide is **4194304 bytes**. This `KMALLOC_MAX_SIZE` constant is defined in the `slab.h` header.

- 4) Is the memory given to you by `kmalloc()` physically contiguous?

As per Stack Overflow Users Ramanau and Yogeesh [2], the `kmalloc()` function guarantees that the pages are both physically and virtually contiguous, although, contiguity for `kmalloc()` may fail if the order of allocation is already very high.

## II. MEMORY AND PAGING (50 PTS)

- 1) Determine the value of the Page Size in the system you are currently using.

Using the code snippet below and printing the kernel module in `dmesg` afterwards:

```
// #1
printf(KERN_INFO "Page size is: %ld
bytes.\n", PAGE_SIZE);
```

```
[ 3928.472193] Page size is: 4096 bytes.
```

Terminal output shows that the value of the page size is **4096 bytes**. This `PAGE_SIZE` constant is also already defined in one of the linux headers.

- 2) Determine the start of the HIGHMEM region of your memory. Clearly state the architecture used by your machine when answering this problem.

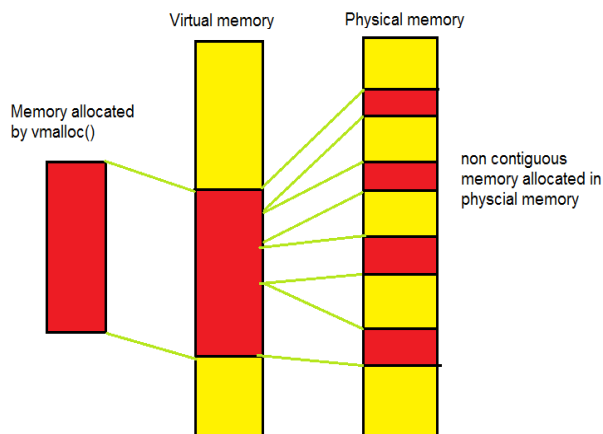
Invoking the `hostnamectl` terminal command, it shows that my system is of the 64-bit architecture.

```
paulinosalmon@paulinosalmon-VirtualBox:~/Desktop/GitClones/C
/Lab/ME7$ hostnamectl
  Static hostname: paulinosalmon-VirtualBox
        Icon name: computer-vm
        Chassis: vm
        Machine ID: 9aa1d72f8e5f401fa8f9cf678e1fbde8
        Boot ID: bcf22be84cd342689190b318db62c746
  Virtualization: oracle
  Operating System: Ubuntu 18.04.3 LTS
        Kernel: Linux 5.0.0-29-generic
  Architecture: x86_64
```

64-bit processors can directly access  $2^{64}$  bytes of memory. As per Users Kelley and wag [3], the linux kernel splits these into the high memory (user space) and the low memory (kernel space). Compared to 32-bit machines in which the high memory address range starts at 0x00000000, high memory does not exist for 64-bit machines as it can access a huge memory of 16 EB, theoretically giving anyone more RAM than they ever need in the entire history of computing.

- 3) Determine if the memory given by `vmalloc()` is physically contiguous per page, and if the memory given by `vmalloc()` is physically contiguous overall.

As per Ramanau and Yogeesh [2], `vmalloc()` allocates memory that is virtually contiguous but not necessarily physically contiguous overall.



- 4) Determine if the memory given by `vmalloc()` in the HIGHMEM region corresponds to a physical memory address.

A memory allocated by `vmalloc()` has a virtual address, but this may not be necessarily mapped directly to a physical address. The `vmalloc()` function only allocates contiguous virtual memory. This entire chunk may be fragmented when it comes to physical memory address translation, as shown in the previous image.

## REFERENCES

- [1] Mike, *Kmalloc size allocation*, Stack Overflow, Accessed 2019-10-06, 2012. [Online]. Available: <https://stackoverflow.com/questions/12568379/kmalloc-size-allocation?fbclid=IwAR1QXv8vtqbtHleLXhgQdwAfsUZgQ8RhoKJUqfOT1TkG5CbqN8k14M>.
- [2] A. Ramanau and H. Yogeesh, *What is the difference between vmalloc and kmalloc?* Stack Overflow, Accessed 2019-10-06, 2017. [Online]. Available: [https://stackoverflow.com/questions/116343/what-is-the-difference-between-vmalloc-and-kmalloc?fbclid=IwAR16xnfAl1nFXua\\_0xKJeKPF885j0tzpfngXanSjavOQ6eZZvUzKjaHGyE8](https://stackoverflow.com/questions/116343/what-is-the-difference-between-vmalloc-and-kmalloc?fbclid=IwAR16xnfAl1nFXua_0xKJeKPF885j0tzpfngXanSjavOQ6eZZvUzKjaHGyE8).
- [3] J. Kelley and wag, *What are high memory and low memory on linux?* Stack Exchange, Accessed 2019-10-06, 2015. [Online]. Available: <https://unix.stackexchange.com/questions/4929/what-are-high-memory-and-low-memory-on-linux>.