

CS342 –Project 3

Report

Arif Can Terzioğlu

21302061

Environment

My main system works on Windows 10 OS. On the hardware side, computer processor is 4 cores Intel i7 4700HQ 2.Ghz series. Ram amount is 16GB.

I installed Ubuntu 16.04 64bit via virtual machine which course assistants provided us. I used kernel version: 4.10.0-28-generic.

Step 1

As project assignment stated, step 1 was a great opportunity get into module concept. It has some obvious differences from regular C program. Modules have a strong connection with kernel version. While I was writing them, I got a lot of problem due to kernel versions. All references and code examples were based on different version of kernel. As kernel version changes, structure of libraries and functions may vary as I understand. For example, at the beginning I tried to use `end_of_stack()` function in kernel 4.13.x. However, its' make file did not complete compilation. However, when I tried it on 4.10, it was successfully done. Also, place of functions can be changed between library files. Therefore; while developing a module, keeping touch with changes on Linux kernel is really necessary. I handled this situation by working on kernel 4.10.0-28 for modules. If we want to select different linux kernel version, we can select another version at the booting screen pressing shift button. Of course, build and run a new kernel version is another option.

Since modules do not permit to use regular C libraries, usage of `printf` function is not possible. I recorded all outputs with `printk` function and display them on the terminal with `dmesg` function. `Printk` function takes different priority parameters such as `KERN_INFO`, `KERN_ALERT` and etc. Also, different than regular C programs main, modules takes `init_module` and `cleanup_module` for initialization and termination respectively.

At the end of step 1, I learnt that modules are used to add new functionality directly into the kernel.

Step 2

The basic usage of modules were learnt in the step 1 such as insmod, object make file, rmmod, dmesg, dmesg -c and etc. This step was more about the memory content and management of the process

Part A

In this part, supplementary notes and fourth reference of the project assignment helped me a lot. Basically what I did is that I create a task_struct then equalize the current task of module. Then starting from that process I iterate over the PCB list until I find the desired "pid" value.

Finding the desired process was very straightforward and fourth reference of the project assignment enabled me to get into memory management part and mm_struct structure.

Part B

Theory

As we learnt in the process chapter of the book, each process divided into some segments such as stack, code, heap, memory mapping segment and bss segment. The virtual memory layout of a process is a list of virtual memory regions. These regions constitute the virtual address space of the process.

As I stated, every process has an associated struct task_struct makes the processes unique in terms of identity and memory management. I want to mention two of member of task_struct that are vm_area_struct and mm_struct.

Each vm_area_struct contain information about a single region in a contiguous range of virtual addresses. An instance of it describes a memory area such as start and end addresses. I used vm_area_struct to find the start and end addresses.

"mm_struct" contains information about general memory layout. In other word, mm_struct provides us to kind of executive summary of memory parts.

Implementation

Mm_struct has a list of vm_area_struct that belongs to the process. I saw the example that iterates through the list and prints the start and end addresses of each in reference four. I used that sample for my code. To find their size, I basically subtract start from end. I asked Ibrahim Hoca, he said probably correct but you should do a more research to be sure. I divided the sizes with page size and I saw they are multiple of page size (4096). In that way I was able to be sure.

When I looked at the detailed structure of the mm_struct on linux kernel v4.10, I realized some unsigned long start addresses name start_code, start_data and etc. I printed the code, stack and all

heap segments via these addresses. However, when I checked the results from `cat /proc/pid/maps`, some of values matched and some of them did not match. More interestingly, addresses are very close to each other but not exactly the same. To match the exact answer, I decided to print each data, stack and other segments through their `vm_areas`. I again iterated over the `vm_area_struct` and compare their start and end address with unsigned long addresses in the `mm_struct`. When I found the correct interval, print them.

Stack was the problematic part in terms of finding its end. Again like other segment, I found it end segment by iterating over the `vm_areas`. Before iteration method, I used method called `end_of_stack(task_struct)`. It did not give the matching result with the `cat /proc/pid/maps`. One confusing point is still each thread's stack regions.

I found the number of frames by using `get_mm_rss` that I saw on the piazza.

At the beginning I supposed that `total_vm` gives the total memory used. However, when I read the piazza I learnt it returns the total number of pages used. Then, I multiply it with page size to get total virtual memory used. However, another parameter of `mm_struct` attracts my attention that is `task_size`. In the source code, with comments it says size of task vm space. So, I gave it a shot to try. However; `task_size` gave really strange numbers that I could not figure out. I am still confused about that parameter. I decided to leave multiplication with `total_vm`. Because at the step 3, increase in heap size was equal to increase in multiplication of `total_vm` and page size.

Some Outputs

```

root@cs342vm: /home/cs342/Desktop/Modules2
root@cs342vm: /home/cs342/Desktop/Modules2# insmod first.ko processId=4620
root@cs342vm: /home/cs342/Desktop/Modules2# dmesg
[3593.793819]
Got the process id to look up as 4620.
[3593.793844] app[4620]
[3593.793846] 0x400000 0x401000
[3593.793847] 0x600000 0x601000
[3593.793848] 0x601000 0x602000
[3593.793849] 0x1bc0000 0x1c39000
[3593.793850] 0x7f1bda7c3000 0x7f1bda983000
[3593.793851] 0x7f1bda983000 0x7f1bda987000
[3593.793852] 0x7f1bda987000 0x7f1bda989000
[3593.793853] 0x7f1bda989000 0x7f1bda98d000
[3593.793854] 0x7f1bda98d000 0x7f1bda9b3000
[3593.793855] 0x7f1bda9b3000 0x7f1bda9b9000
[3593.793856] 0x7f1bda9b9000 0x7f1bda9b3000
[3593.793857] 0x7f1bda9b3000 0x7f1bda9b4000
[3593.793858] 0x7f1bda9b4000 0x7f1bda9b5000
[3593.793859] 0x7ffff53fde000 0x7ffff53ffff000
[3593.793860] 0x7ffff5408000 0x7ffff5408000
[3593.793861] 0x7ffff5408000 0x7ffff5408f000
[3593.793862]
Code Segment start = 0x400000, end = 0x400914, size 0x2324
Data Segment start = 0x600e10, end = 0x601058, size 0x584
Stack Segment start = 0x7ffff53ffc960, end = 0xfffffa49c84324000
size = 0x18445517714051298560
Heap Segment start = 0x1bc0000, end = 0x1c39000, size 0x495616
Arg Segment start = 0x7ffff53ffe22f, end = 0x7ffff53ffe235, size 0x6
Env Segment start = 0x7ffff53ffe235, end = 0x7ffff53ffeff2, size 0x3517
VM Number of Frames = 0x194
VM size = 0x140737488351232
VM size2 = 0x4812800
root@cs342vm: /home/cs342/Desktop/Modules2#

```

```

root@cs342vm: /home/cs342/Desktop/Step2
[28240.654378] 1 1 1 0 0 1 1 0 0 644108 0 0
[37094.939155]
Print segment information module exiting.
root@cs342vm: /home/cs342/Desktop/Step2# clear
root@cs342vm: /home/cs342/Desktop/Step2# dmesg-
No command 'dmesg-' found, did you mean:
Command 'dmesg' from package 'util-linux' (main)
dmesg+: command not found
root@cs342vm: /home/cs342/Desktop/Step2# dmesg
root@cs342vm: /home/cs342/Desktop/Step2# insmod module1.ko processId=23451
root@cs342vm: /home/cs342/Desktop/Step2# dmesg
[37139.359615]
Got the process id to look up as 23451.
[37139.359668] app[23451]
[37139.359670]
Code Segment start = 0x400000, end = 0x401000
size = 4096
[37139.359671]
Data Segment start = 0x600000, end = 0x601000
size = 4096
[37139.359673]
Stack Segment start = 0x7ffff3a95c000, end = 0x7ffff3a97d000
size = 135168
[37139.359674]
Heap Segment start = 0x2301000, end = 0x2322000, size 135168
[37139.359675]
Arg Segment start = 0x7ffff3a95c000, end = 0x7ffff3a97d000
size = 135168
[37139.359676]
Env Segment start = 0x7ffff3a95c000, end = 0x7ffff3a97d000
size = 135168
[37139.359677]
VM Number of Frames = 0x159
[37139.359678]
VM size = 0x140737488351232
[37139.359679]
VM size = 0x4452352
[37139.359680] a358067
[37139.359682] 1 1 1 0 0 1 1 0 0 670240 0 0
[37139.359683] a38b3067
[37139.359685] 1 1 1 0 0 1 1 0 0 669875 0 0
[37139.359685] cb740067
[37139.359687] 1 1 1 0 0 1 1 0 0 833376 0 0
[37139.359688] 9da28067
[37139.359689] 1 1 1 0 0 1 1 0 0 645672 0 0
[37139.359690] 11d0a4067
[37139.359692] 1 1 1 0 0 1 1 0 0 1167524 0 0
[37139.359693] 110472067
[37139.359694] 1 1 1 0 0 1 1 0 0 1115250 0 0
[37139.359695] 11d852067
[37139.359697] 1 1 1 0 0 1 1 0 0 1169490 0 0
[37139.359697] 9d74d067
[37139.359699] 1 1 1 0 0 1 1 0 0 645085 0 0
[37139.359700] 9d4bc067
[37139.359701] 1 1 1 0 0 1 1 0 0 644108 0 0
root@cs342vm: /home/cs342/Desktop/Step2#

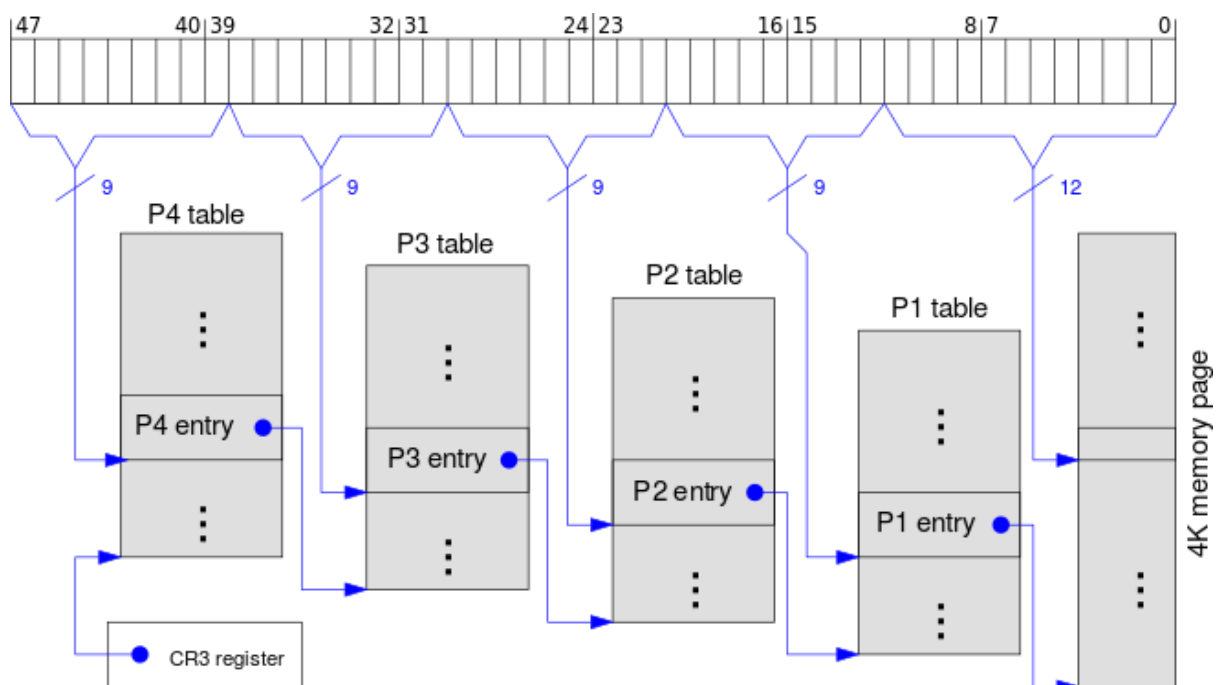
```

The first VM size is with task_size.

Part C

Theory and Implementation

I started to this part by observing the four level paging structures. Since we are only interested in only top level page table, first 9 bit of our 48 bit virtual address is our concern. While I was thinking about how can I parse these 9-bits and reach the top level page table. I learnt from reference 7 that `mm_struct` has a pointer `pgd` that points to the top level page table. In this table, each entry corresponds to second level page table.



This image is taken from the [page table](#) that some students share on the piazza. That links helped me a lot in terms of understanding. I investigated some approaches on the piazza about this part. Some methods use `pgd_offset` function to get the table entries. As far as I understand, we need to increase the address by 2^{39} to increase PGD address (2^{48} unique address / 2^9 top level pages). From this point, we can understand, we can iterate over `pgd_t` instance to access the each top level page. Due to offset we have 512 entries. For getting the each page entry I used `pgd_val()` function which is easier than offset function. I saw that function in supplementary notes. I printed each of entry that's valid bit is not 0. Basically, I iterated over all `pgd` entries until 511 as an index, get their page table entry value then print it.

Next stage was parsing. To understand what I need to parse, I had to talk and read a lot. This discussion was an ambiguity among our friends also. Although Intel document seems very

straightforward, to understand what these bits or bit was hard stage. I shifted and masked the entries to get the desired bits.

As a result, I am able to print these bits on the Intel document Table 4-14.

Present bit 0: If entry is used it must be 1.

Read/Write 1: If 0, writes may not be allowed and etc.

I learnt each of these bits and bit meaning from the document.

Some Outputs

```
root@cs342vm: /home/cs342/Desktop/Step2
root@cs342vm: /home/cs342/Desktop/Step2# insmod module1.ko processId=23990
root@cs342vm: /home/cs342/Desktop/Step2# dmesg
[43468.086023] Got the process id to look up as 23990.
[43468.086049] codeblocks[23990]
[43468.086051] Code Segment start = 0x400000 end = 0x5aa000
size = 1744896
[43468.086053] Data Segment start = 0x7a9000 end = 0x7bd000
size = 81920
[43468.086090] Stack Segment start = 0x7ffc5838d000 end = 0x7ffc583ae000
size = 135168
[43468.086099] Heap Segment start = 0x17b9000 end = 0x48bb000 size 51388416
[43468.086107] Arg Segment start = 0x7ffc5838d000 end = 0x7ffc583ae000
size = 135168
[43468.086115] Env Segment start = 0x7ffc5838d000 end = 0x7ffc583ae000
size = 135168
[43468.086110] VM Number of Frames = 0x25269
[43468.086117] VM size = 0x140737488351232
[43468.086118] VM size = 0x1012322304
[43468.086119] 811c0067 1 1 1 0 0 1 1 0 0 528032 0 0
[43468.086121] 798ed067 1 1 1 0 0 1 1 0 0 497901 0 0
[43468.086124] 1 1 1 0 0 1 1 0 0 529321 0 0
[43468.086126] 9da28067 1 1 1 0 0 1 1 0 0 645672 0 0
[43468.086127] 110a4067 1 1 1 0 0 1 1 0 0 1167524 0 0
[43468.086128] 110a4067 1 1 1 0 0 1 1 0 0 1115250 0 0
[43468.086131] 11d852067 1 1 1 0 0 1 1 0 0 1169490 0 0
[43468.086132] 9d7dd067 1 1 1 0 0 1 1 0 0 645085 0 0
[43468.086133] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
[43468.086134] 11d852067 1 1 1 0 0 1 1 0 0 644108 0 0
[43468.086135] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
[43468.086136] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
[43468.086137] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
[43468.086138] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
[43468.086139] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
[43468.086140] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
root@cs342vm: /home/cs342/Desktop/Step2#
```

```
root@cs342vm: /home/cs342/Desktop/Step2
[37139.359695] 11d852067 1 1 1 0 0 1 1 0 0 1169490 0 0
[37139.359697] 9d7dd067 1 1 1 0 0 1 1 0 0 645085 0 0
[37139.359699] 1 1 1 0 0 1 1 0 0 645085 0 0
[37139.359700] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
[37139.359701] 1 1 1 0 0 1 1 0 0 644108 0 0
[43304.875367] Print segment information module exiting.
root@cs342vm: /home/cs342/Desktop/Step2# clear
root@cs342vm: /home/cs342/Desktop/Step2# insmod module1.ko processId=23859
root@cs342vm: /home/cs342/Desktop/Step2# dmesg
[43327.000807] Got the process id to look up as 23859.
[43327.000833] app[23859]
[43327.000835] Code Segment start = 0x400000 end = 0x401000
size = 4096
[43327.000836] Data Segment start = 0x600000 end = 0x601000
size = 4096
[43327.000838] Stack Segment start = 0x7ffe01778000 end = 0x7ffe01799000
size = 135168
[43327.000839] Heap Segment start = 0xc28000 end = 0xc49000 size 135168
[43327.000841] Arg Segment start = 0x7ffe01778000 end = 0x7ffe01799000
size = 135168
[43327.000842] Env Segment start = 0x7ffe01778000 end = 0x7ffe01799000
size = 135168
[43327.000843] VM Number of Frames = 0x161
[43327.000843] VM size = 0x140737488351232
[43327.000844] VM size = 0x4452352
[43327.000845] cob1b067 1 1 1 0 0 1 1 0 0 789275 0 0
[43327.000847] cead3067 1 1 1 0 0 1 1 0 0 789203 0 0
[43327.000850] 1 1 1 0 0 1 1 0 0 677851 0 0
[43327.000851] 257d0067 1 1 1 0 0 1 1 0 0 645672 0 0
[43327.000852] 9da28067 1 1 1 0 0 1 1 0 0 1167524 0 0
[43327.000853] 110a4067 1 1 1 0 0 1 1 0 0 1115250 0 0
[43327.000857] 11d852067 1 1 1 0 0 1 1 0 0 1169490 0 0
[43327.000858] 110a4067 1 1 1 0 0 1 1 0 0 1115250 0 0
[43327.000861] 11d852067 1 1 1 0 0 1 1 0 0 1169490 0 0
[43327.000862] 1 1 1 0 0 1 1 0 0 645085 0 0
[43327.000863] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
[43327.000865] 1 1 1 0 0 1 1 0 0 645085 0 0
[43327.000867] 9d7dd067 1 1 1 0 0 1 1 0 0 644108 0 0
root@cs342vm: /home/cs342/Desktop/Step2#
```

Step 3

Heap Allocation and Deallocation

Input Allocation sizes: 120KB, 180KB, 300KB, 360KB and 420KB

Note: For each allocation, program is started newly. I mean increases are calculated based on initial size.

It needs to point out that we need to allocate memory from heap part by part. I mean as stated in the piazza, "if you call malloc() with a request higher than 135KB, then it will call mmap() systemcall and satisfy your request by mapping ANOTHER memory region outside of heap. Thus, heap will not enlarge."

Thus, to enlarge the heap, I used sequence of 60KB allocations.

Results

I allocated memory with "malloc" function with given sizes above. Our initial heap size is around 135KB.

Exact number is 135168 byte. Important point is that this size is multiple of our page size that is $135168 / 4096 = 33$. It means we have 33 pages initially.

Our initial vm space size is 4,452,352 that is 1087 pages total.

When we allocate 120KB initially, since current heap is enough, heap was not enlarged.

When we allocate 180KB, our heap size becomes 315392 byte. Increase is 180224 byte 44 pages in both total size and heap size. We can see that increase equals to allocation amount.

If we allocate 300KB, sizes does not change.

When we allocate 360KB, our heap size becomes 495616 byte. Increase is 360448 byte 88 pages in both total size and heap size.

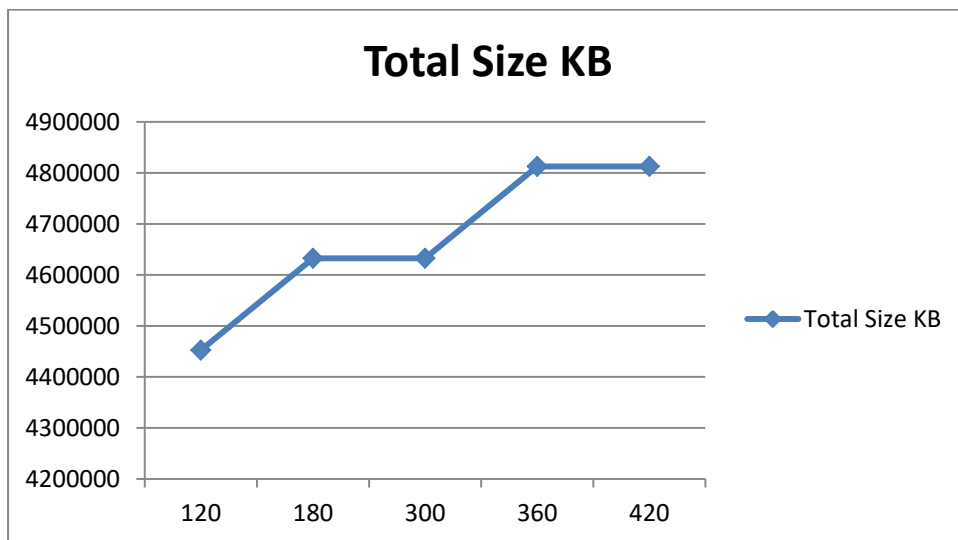
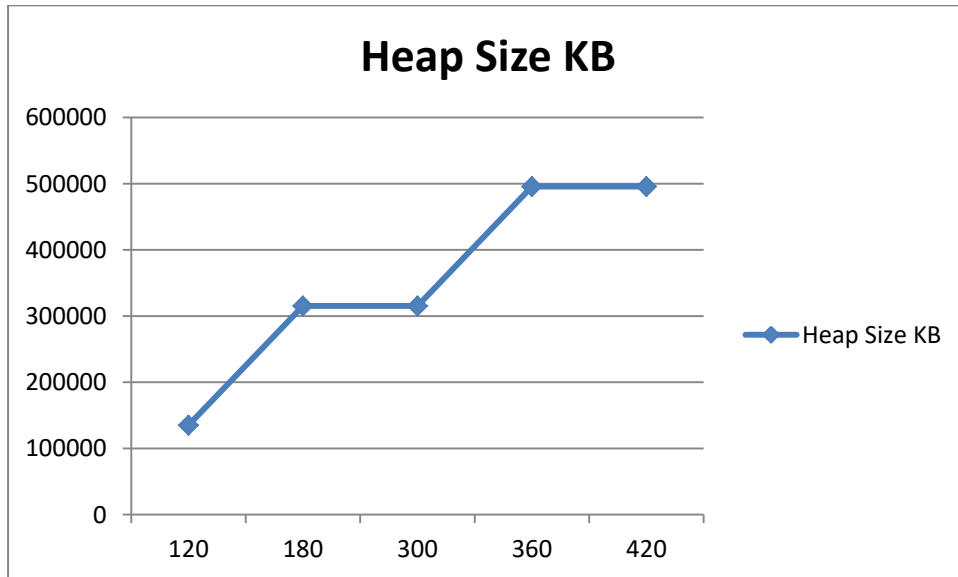
If we allocate 420KB, sizes are the same with 360KB. Changes did not occur my guess is our current heap size in 360 KB 495616 which is also enough for 420KB.

Number of page count increase equals to number of page count increase in total size.

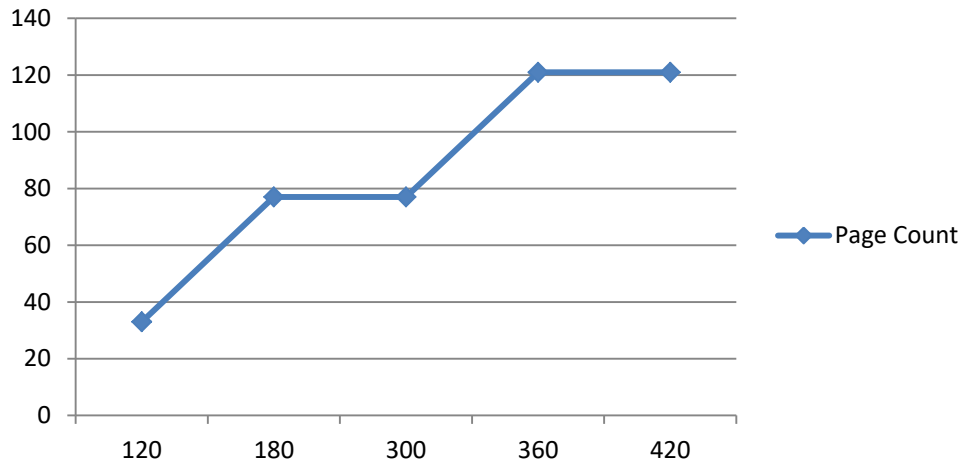
Deallocation

On the deallocation part, when we deallocate the memory heap turned to its initial size that is 135KB. In the experiment, when I deallocate all of the heap areas i have allocated, at the end the heap decreases back to its default size.

Graphics



Page Count Stack



Some Outputs

```
root@cs342vm: /home/cs342/Desktop/Step2
cs342@cs342vm:~/Desktop/Step2$ sudo su
[sudo] password for cs342:
root@cs342vm: /home/cs342/Desktop/Step2# insmod module1.ko processId=5082
root@cs342vm: /home/cs342/Desktop/Step2# dmesg
0093.050285] Got the process Id to look up as 5082.
0093.050312] app[5082]
0093.050314] Code Segment start = 0x400000 end = 0x401000 size = 4096
0093.050315] Data Segment start = 0x600000 end = 0x601000 size = 4096
0093.050317] Stack Segment start = 0x7fff7681b000 end = 0x7fff7683c000 size = 135168
0093.050318] Heap Segment start = 0x84c000 end = 0x86d000 size 135168
0093.050319] Arg Segment start = 0x7fff7681b000 end = 0x7fff7683c000 size = 135168
0093.050321] Env Segment start = 0x7fff7681b000 end = 0x7fff7683c000 size = 135168
0093.050321] VM Number of Frames = 0x194
0093.050322] Task size = 0x140737488351232
0093.050323] VM size = 0x4452352
0093.050324] 846d3067
0093.050326] 1 1 1 0 0 1 1 0 0 542419 0 0
0093.050327] 846d4067
0093.050329] 1 1 1 0 0 1 1 0 0 542340 0 0
0093.050330] 8d228067
0093.050334] 1 1 1 0 0 1 1 0 0 578088 0 0
0093.050336] 11d0a4067
0093.050337] 1 1 1 0 0 1 1 0 0 1167524 0 0
0093.050338] 1104a067
0093.050339] 1 1 1 0 0 1 1 0 0 1115166 0 0
0093.050340] 11d852067
0093.050342] 1 1 1 0 0 1 1 0 0 1169490 0 0
0093.050343] 8cfd067
0093.050344] 1 1 1 0 0 1 1 0 0 577501 0 0
0093.050345] 8cc0c067
0093.050347] 1 1 1 0 0 1 1 0 0 576524 0 0
root@cs342vm: /home/cs342/Desktop/Step2#
```

```
root@cs342vm: /home/cs342/Desktop/Step2$ sudo su
cs342@cs342vm:~/Desktop/Step2$ sudo su
[sudo] password for cs342:
root@cs342vm: /home/cs342/Desktop/Step2# insmod module1.ko processId=5254
root@cs342vm: /home/cs342/Desktop/Step2# dmesg
0946.363884] Got the process id to look up as 5254.
0946.363890] app[5254]
0946.363892] Code Segment start = 0x400000 end = 0x401000 size = 4096
0946.363893] Data Segment start = 0x600000 end = 0x601000 size = 4096
0946.363895] Stack Segment start = 0x7ffc00d9f000 end = 0x7ffc00dc0000 size = 135168
0946.363897] Heap Segment start = 0x24e1000, end = 0x255a000, size 495616
0946.363898] Arg Segment start = 0x7ffc00d9f000 end = 0x7ffc00dc0000 size = 135168
0946.363899] Env Segment start = 0x7ffc00d9f000 end = 0x7ffc00dc0000 size = 135168
0946.363900] VM Number of Frames = 0x156
0946.363901] Task size = 0x140737488351232
0946.363902] VM size = 0x4812800
0946.363903] 97840067
0946.363904] 1 1 1 0 0 1 1 0 0 620608 0 0
0946.363905] 97840067
0946.363906] 1 1 1 0 0 1 1 0 0 620612 0 0
0946.363907] 97815067
0946.363908] 1 1 1 0 0 1 1 0 0 620565 0 0
0946.363909] 8d228067
0946.363910] 1 1 1 0 0 1 1 0 0 578088 0 0
0946.363911] 11d0a4067
0946.363912] 1 1 1 0 0 1 1 0 0 1167524 0 0
0946.363913] 11041e067
0946.363914] 1 1 1 0 0 1 1 0 0 115166 0 0
0946.363915] 11d852067
0946.363916] 1 1 1 0 0 1 1 0 0 1169490 0 0
0946.363917] 8cfd067
0946.363918] 1 1 1 0 0 1 1 0 0 577501 0 0
0946.363919] 8cc0c067
0946.363920] 1 1 1 0 0 1 1 0 0 576524 0 0
root@cs342vm: /home/cs342/Desktop/Step2#
```

```
root@cs342vm: /home/cs342/Desktop/Step2$ sudo su
cs342@cs342vm:~/Desktop/Step2$ sudo su
[sudo] password for cs342:
root@cs342vm: /home/cs342/Desktop/Step2# insmod module1.ko processId=5191
root@cs342vm: /home/cs342/Desktop/Step2# dmesg
0816.293506] Got the process id to look up as 5191.
0816.293511] app[5191]
0816.293513] Code Segment start = 0x400000 end = 0x401000 size = 4096
0816.293514] Data Segment start = 0x600000 end = 0x601000 size = 4096
0816.293515] Stack Segment start = 0x7ffd5bce9000 end = 0x7ffd5bd0a000 size = 135168
0816.293516] Heap Segment start = 0x2226c000, end = 0x22b90000, size 315392
0816.293517] Arg Segment start = 0x7ffd5bce9000 end = 0x7ffd5bd0a000 size = 135168
0816.293518] Env Segment start = 0x7ffd5bce9000 end = 0x7ffd5bd0a000 size = 135168
0816.293519] VM Number of Frames = 0x158
0816.293520] Task size = 0x140737488351232
0816.293521] VM size = 0x4632576
0816.293522] c6e08067
0816.293523] 1 1 1 0 0 1 1 0 0 814600 0 0
0816.293524] c23cd067
0816.293525] 1 1 1 0 0 1 1 0 0 795597 0 0
0816.293526] c6f0b067
0816.293527] 1 1 1 0 0 1 1 0 0 815019 0 0
0816.293528] 8d228067
0816.293529] 1 1 1 0 0 1 1 0 0 578088 0 0
0816.293530] 11d0a4067
0816.293531] 1 1 1 0 0 1 1 0 0 1167524 0 0
0816.293532] 11041e067
0816.293533] 1 1 1 0 0 1 1 0 0 115166 0 0
0816.293534] 11d852067
0816.293535] 1 1 1 0 0 1 1 0 0 1169490 0 0
0816.293536] 8cfd067
0816.293537] 1 1 1 0 0 1 1 0 0 577501 0 0
0816.293538] 8cc0c067
0816.293539] 1 1 1 0 0 1 1 0 0 576524 0 0
root@cs342vm: /home/cs342/Desktop/Step2#
```

Stack Allocation and Deallocation

During the trial of stack allocation I mean recursive function, I tried Fibonacci but I could not increase the heap size. Then, I turned to factorial.

Note: I could not decrease the stack after recursion. As it stated in piazza, maybe the compiler predetermined at compile time for optimization purposes.

Results

Observation: In heap allocation, only total size and heap size was increased. However, in stack allocation, with stack size also number of frames, environment variable size and main argument sizes are also increases parallel to stack size. It can be occurred that since recursive function uses new parameters and arguments in each recursion, these parameters are also extended. Therefore, total size in stack increases more than heap increases.

As in heap size, stack initial size is also 135KB.

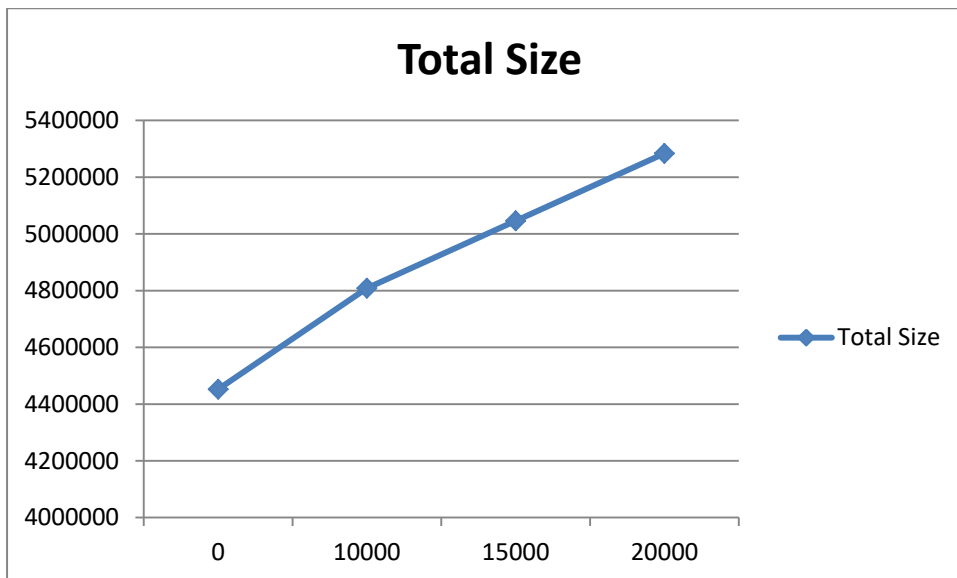
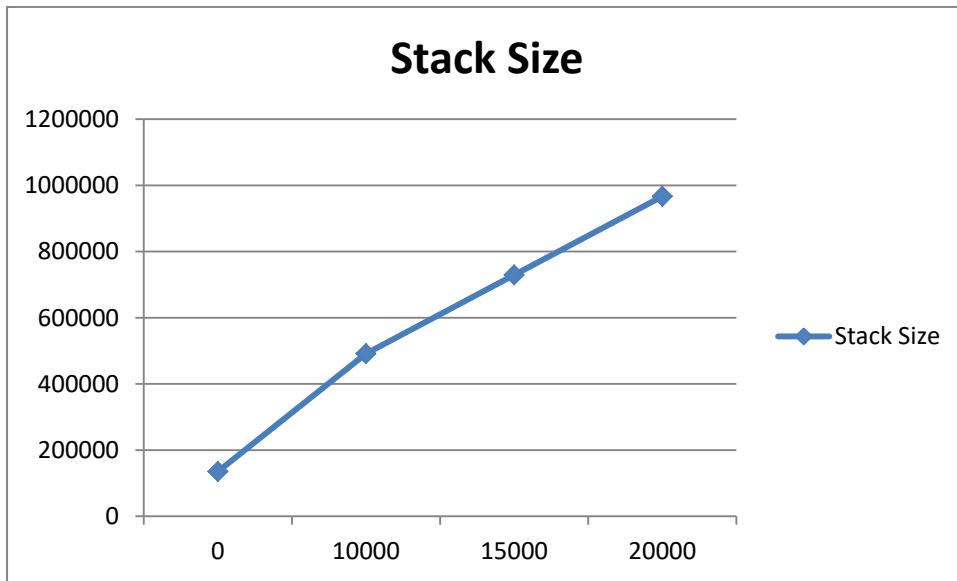
Exact number is 135168 byte. Important point is again that this size is multiple of our page size that is $135168 / 4096 = 33$. It means we have 33 pages initially.

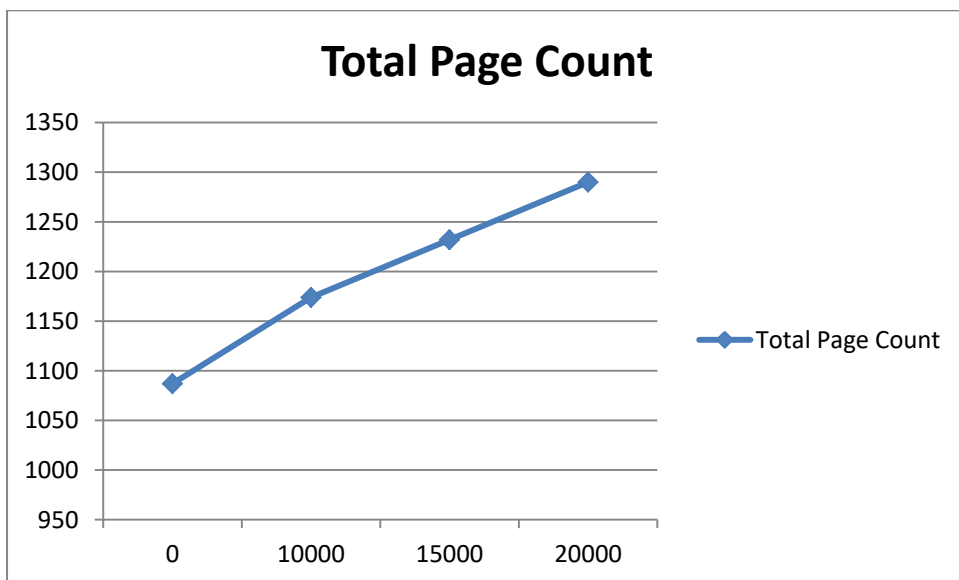
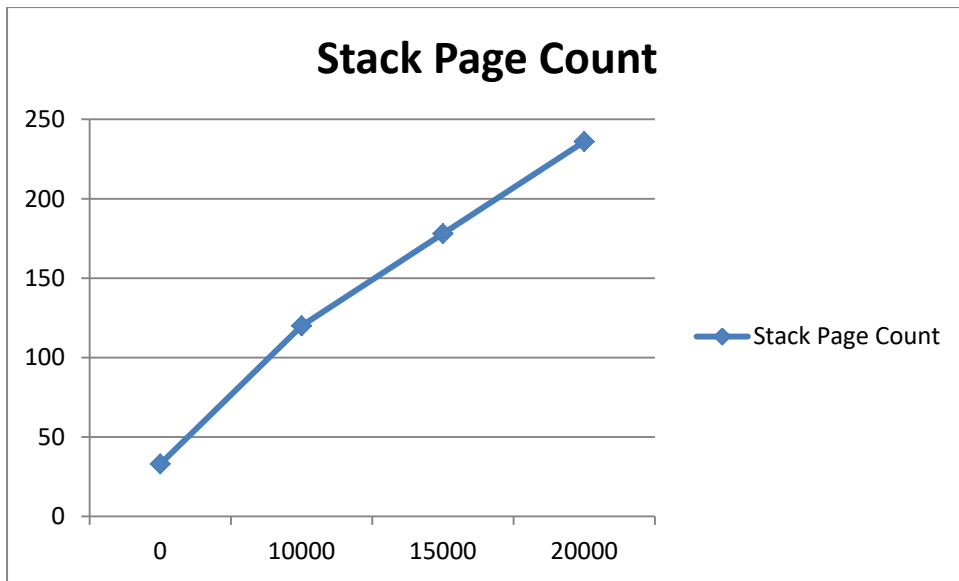
When we calculate 10000!, our stack size becomes 491520 that is 120 pages. In total we have 4452352 bytes with 1087 pages. One point is that increase in sizes is kind of larger than increase in heap. As I stated, stack is not only parameter that increases. Variable, argument and frames are also increasing.

When we calculate 15000!, our stack size becomes 729088 that is 178 pages. In total we have 5046272 bytes with 1174 pages.

When we calculate 20000!, our stack size becomes 966656 that is 236 pages. In total we have 5283840 bytes with 1290 pages. We can see that in heap increase in input may not be resulted in increases in heap size. However, in here I get always increase in size. It may be occurred from that I kept interval between two inputs very large.

Graphics





Some Outputs

```
root@cs342vm: /home/cs342/Desktop/Step2
cs342@cs342vm:~/Desktop/Step2$ sudo su
[sudo] password for cs342:
root@cs342vm: /home/cs342/Desktop/Step2$ insmod module1.ko processId=6779
root@cs342vm: /home/cs342/Desktop/Step2$ dmesg
[12654.616148] Got the process id to look up as 6779.
[12654.616172] app[6779]
[12654.616174] Code Segment start = 0x400000 end = 0x401000 size = 4096
[12654.616176] Data Segment start = 0x600000 end = 0x601000 size = 4096
[12654.616178] Stack Segment start = 0x7fffd81b50000 end = 0x7fffd81b71000 size = 135168
[12654.616179] Heap Segment start = 0x76e000, end = 0x78f000, size 135168
[12654.616180] Arg Segment start = 0x7fffd81b50000 end = 0x7fffd81b71000 size = 135168
[12654.616181] Env Segment start = 0x7fffd81b50000 end = 0x7fffd81b71000 size = 135168
[12654.616182] VM Number of Frames = 0x156
[12654.616183] Task size = 0x140737488351232
[12654.616184] VM size = 0x4452352
[12654.616185] d9d1e067
[12654.616187] 1 1 1 0 0 1 1 0 0 892182 0 0
[12654.616188] 09100067
[12654.616190] 1 1 1 0 0 1 1 0 0 430336 0 0
[12654.616191] 8d228067
[12654.616192] 1 1 1 0 0 1 1 0 0 578088 0 0
[12654.616193] 1108a4067
[12654.616195] 1 1 1 0 0 1 1 0 0 1167524 0 0
[12654.616196] 11041e067
[12654.616197] 1 1 1 0 0 1 1 0 0 1115166 0 0
[12654.616198] 11d852067
[12654.616200] 1 1 1 0 0 1 1 0 0 1169490 0 0
[12654.616201] 8cfd4067
[12654.616202] 1 1 1 0 0 1 1 0 0 577501 0 0
[12654.616203] 8cc0c067
[12654.616204] 1 1 1 0 0 1 1 0 0 576524 0 0
root@cs342vm: /home/cs342/Desktop/Step2$
```

```
root@cs342vm: /home/cs342/Desktop/Step2
cs342@cs342vm:~/Desktop/Step2$ sudo su
[sudo] password for cs342:
root@cs342vm: /home/cs342/Desktop/Step2$ insmod module1.ko processId=7032
root@cs342vm: /home/cs342/Desktop/Step2$ dmesg
[13148.826121] Got the process id to look up as 7032.
[13148.826140] app[7032]
[13148.826149] Code Segment start = 0x400000 end = 0x401000 size = 4096
[13148.826151] Data Segment start = 0x600000 end = 0x601000 size = 4096
[13148.826152] Stack Segment start = 0x7fffe78b9000 end = 0x7fffe7931000 size = 491520
[13148.826153] Heap Segment start = 0x1471000, end = 0x1492000, size 135168
[13148.826155] Arg Segment start = 0x7fffe78b9000 end = 0x7fffe7931000 size = 491520
[13148.826155] Env Segment start = 0x7fffe78b9000 end = 0x7fffe7931000 size = 491520
[13148.826156] VM Number of Frames = 0x400
[13148.826157] Task size = 0x140737488351232
[13148.826157] VM size = 0x4808704
[13148.826158] c2372067
[13148.826160] 1 1 1 0 0 1 1 0 0 795506 0 0
[13148.826162] c6f9c067
[13148.826163] 1 1 1 0 0 1 1 0 0 815004 0 0
[13148.826164] 97a3e067
[13148.826166] 1 1 1 0 0 1 1 0 0 621118 0 0
[13148.826168] 8d228067
[13148.826169] 1 1 1 0 0 1 1 0 0 578088 0 0
[13148.826169] 1108a4067
[13148.826171] 1 1 1 0 0 1 1 0 0 1167524 0 0
[13148.826171] 11041e067
[13148.826173] 1 1 1 0 0 1 1 0 0 1115166 0 0
[13148.826174] 11d852067
[13148.826175] 1 1 1 0 0 1 1 0 0 1169490 0 0
[13148.826176] 8cfd4067
[13148.826178] 1 1 1 0 0 1 1 0 0 577501 0 0
[13148.826178] 8cc0c067
[13148.826180] 1 1 1 0 0 1 1 0 0 576524 0 0
root@cs342vm: /home/cs342/Desktop/Step2$
```

```
root@cs342vm: /home/cs342/Desktop/Step2
cs342@cs342vm:~/Desktop/Step2$ sudo su
[sudo] password for cs342:
root@cs342vm: /home/cs342/Desktop/Step2# insmod module1.ko processId=6962
root@cs342vm: /home/cs342/Desktop/Step2# dmesg
[12992.625600]
Got the process id to look up as 6962.
[12992.625615] app[6962]
[12992.625618]
Code Segment start = 0x400000 end = 0x401000 size = 4096
[12992.625618]
Data Segment start = 0x600000 end = 0x601000 size = 4096
[12992.625620]
Stack Segment start = 0x7ffc55acf000 end = 0x7ffc55bbb000 size = 966656
[12992.625623]
Heap Segment start = 0x7fb000, end = 0x79c000, size 135168
[12992.625622] Arg Segment start = 0x7ffc55acf000 end = 0x7ffc55bbb000 size = 966656
[12992.625623] Env Segment start = 0x7ffc55acf000 end = 0x7ffc55bbb000 size = 966656
[12992.625624]
VM Number of Frames = 0x538
[12992.625625]
Task size = 0x140737488351232
[12992.625626] VM size = 0x5283840
[12992.625627] c6f5d067
[12992.625628] 1 1 1 0 0 1 1 0 0 814941 0 0
[12992.625630] 8475d067
[12992.625632] 1 1 1 0 0 1 1 0 0 542550 0 0
[12992.625632] 8d228067
[12992.625634] 1 1 1 0 0 1 1 0 0 578088 0 0
[12992.625635] 11d0a4067
[12992.625637] 1 1 1 0 0 1 1 0 0 1167524 0 0
[12992.625637] 11041e067
[12992.625639] 1 1 1 0 0 1 1 0 0 1115166 0 0
[12992.625640] 11d852067
[12992.625641] 1 1 1 0 0 1 1 0 0 1169490 0 0
[12992.625642] 8cfd067
[12992.625644] 1 1 1 0 0 1 1 0 0 577501 0 0
[12992.625644] 8cc0c067
[12992.625646] 1 1 1 0 0 1 1 0 0 576524 0 0
root@cs342vm: /home/cs342/Desktop/Step2#
```

```
root@cs342vm: /home/cs342/Desktop/Step2
cs342@cs342vm:~/Desktop/Step2$ sudo su
[sudo] password for cs342:
root@cs342vm: /home/cs342/Desktop/Step2# insmod module1.ko processId=7145
root@cs342vm: /home/cs342/Desktop/Step2# dmesg
[13212.624062]
Got the process id to look up as 7145.
[13212.624068] app[7145]
[13212.624070]
Code Segment start = 0x400000 end = 0x401000 size = 4096
[13212.624072]
Data Segment start = 0x600000 end = 0x601000 size = 4096
[13212.624074]
Stack Segment start = 0x7ffffdf849000 end = 0x7ffffdf8fb000 size = 729088
[13212.624075]
Heap Segment start = 0x12b2000, end = 0x12d3000, size 135168
[13212.624076] Arg Segment start = 0x7ffffdf849000 end = 0x7ffffdf8fb000 size = 729088
[13212.624077] Env Segment start = 0x7ffffdf849000 end = 0x7ffffdf8fb000 size = 729088
[13212.624078]
VM Number of Frames = 0x471
[13212.624079]
Task size = 0x140737488351232
[13212.624080] VM size = 0x5046272
[13212.624081] 97846067
[13212.624083] 1 1 1 0 0 1 1 0 0 620614 0 0
[13212.624084] aad06067
[13212.624086] 1 1 1 0 0 1 1 0 0 699782 0 0
[13212.624087] 8d228067
[13212.624088] 1 1 1 0 0 1 1 0 0 578088 0 0
[13212.624089] 11d0a4067
[13212.624091] 1 1 1 0 0 1 1 0 0 1167524 0 0
[13212.624092] 11041e067
[13212.624094] 1 1 1 0 0 1 1 0 0 1115166 0 0
[13212.624094] 11d852067
[13212.624096] 1 1 1 0 0 1 1 0 0 1169490 0 0
[13212.624097] 8cfd067
[13212.624099] 1 1 1 0 0 1 1 0 0 577501 0 0
[13212.624099] 8cc0c067
[13212.624091] 1 1 1 0 0 1 1 0 0 576524 0 0
root@cs342vm: /home/cs342/Desktop/Step2#
```