

Part 2:

Basic terminology:

kmalloc() memory allocation: kmalloc() function allocates the space in the physical memory in a contiguous location. It allocates a chunk of memory in the kernel space during the allocation.

vmalloc()memory allocation: Whereas the vmalloc() memory allocation is non-contiguous. Unlike kmalloc it doesn't allocates the memory the contiguous location in the kernel address space. vmalloc() is needed when we ask for large chunk of data block.

Our aim is to find the physical address corresponds to kmalloc and vmalloc allocation using our kernel loadable module.

When we allocate a memory chunk using kmalloc it gets load in a contiguous memory location of kmalloc virtual memory space of kernel. Now as the kmalloc allocations are contiguous in nature, hence the macro present within the kernel like _pa(virtual address) can easily able to find the corresponding physical address by subtracting the offset value from the physical address. Also it is possible to find the physical address of the kmalloc address space using the software walk from the kernel page table.

But in case of vmalloc macros which subtract the offset vale cannot provide us the correct physical address as the vmalloc stores the data in non-contiguous memory location. But its is possible to find the physical address from the kernel page table by performing the software page walk.

Another important concept here is that the entire kernel processes shares the same page table structure. Hence if we allocate memory and using kmalloc or vmalloc in a kernel process we can access that memory location in any other kernel process using the address of the location.

Results and Experiment:

Kmalloc results:

We are creating a structure in the kernel module present in codeDir/Part2/part1/p2_1.c. Then we try to get the physical address of the structure memory allocated using kmalloc.

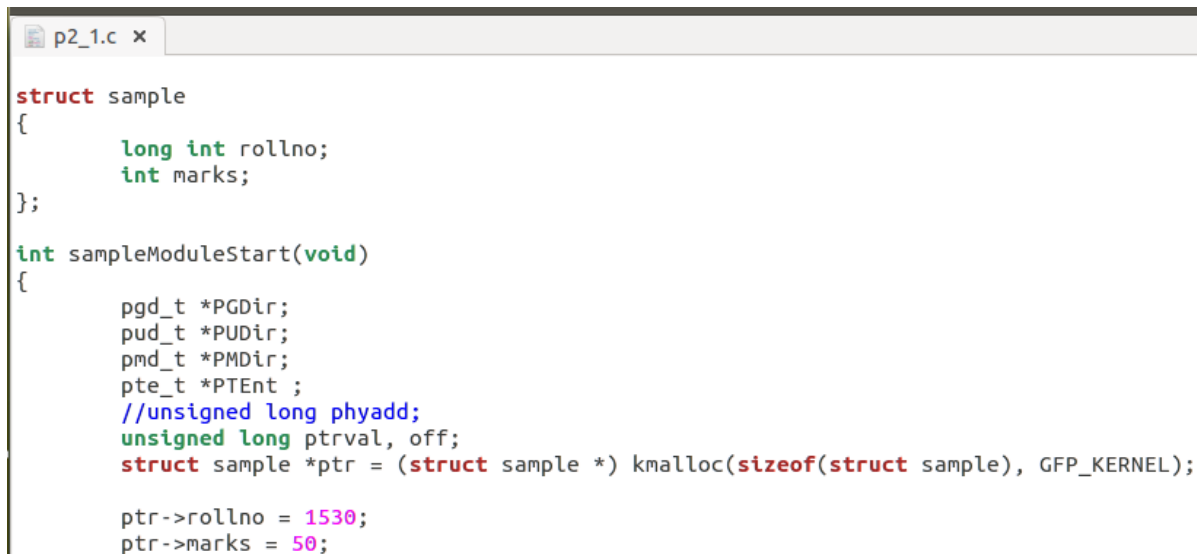
Both the macro and the software page walk physical address are same and giving the correct result as the kmalloc allocates the memory contiguously.

Now we try to do the same again from another module /codeDir/Part2/part1/p2_2.c and also try to access the data of the structure from the new module while the first module is still loaded. The results are still same.

Values given in the first module:

```
[ 100.795289] Address to structure: 18446612134773645872
[ 100.795294] Structure: Virtual address : ffff88009297e230 and Physical address : 9297e230
[ 100.795297] Form software walk: Structure Virtual address : ffff88009297e230 and Physical address : 9297e230
```

Data given in the 1st module:



```
p2_1.c x
struct sample
{
    long int rollno;
    int marks;
};

int sampleModuleStart(void)
{
    pgd_t *PGDir;
    pud_t *PUDir;
    pmd_t *PMDir;
    pte_t *PTEnt ;
    //unsigned long phyadd;
    unsigned long ptrval, off;
    struct sample *ptr = (struct sample *) kmalloc(sizeof(struct sample), GFP_KERNEL);

    ptr->rollno = 1530;
    ptr->marks = 50;
```

Output from the 2nd module:

```
[ 208.412546] Values are roll no = 1530 and marks = 50
[ 208.412556] Structure: Virtual address : ffff88009297e230 and Physical address : 9297e230
[ 208.412561] Form software walk: Structure Virtual address : ffff88009297e230 and Physical address : 9297e230
[ 225.605772] *****Exiting Module*****
```

vmalloc results:

We are creating a structure in the kernel module present in codeDir/Part2/part3/p2_3.c. Then we try to get the physical address of the structure memory allocated using vmalloc.

The macro and the software page walk physical address will not give the same result here. Infact as mentioned earlier, as the macros only subtract the offset from the virtual address to get the physical address, this methods works on kmalloc, as it allocates memory in the contiguous location. But this is not the case in vmalloc, as its not contiguous.

Now we try to do the same again from another module /codeDir/Part2/part4/p2_4.c and also try to access the data of the structure from the new module while the first module is still loaded. The results are still same. Also it can access the values of the structure loaded by the 1st module.

Values given in the first module:

```
root@bubu:/home/suhit/cs746/Part3# insmod p2_3.ko
root@bubu:/home/suhit/cs746/Part3# dmesg | tail -3
[ 375.825710] Address to structure: 18446683600846671872
[ 375.825718] Structure: Virtual address : ffffc900107d5000 and Physical address : 4100107d5000
[ 375.825722] Form software walk: Structure Virtual address : ffffc900107d5000 and Physical address : 13f5a6000
```

As it can be seen, the results using the macros are not correct.

Data given in the 1st module:

Data access from 2nd module and the address mapping:

```
[ 419.263934] Values are roll no = 1530 and marks = 50
[ 419.263943] Structure: Virtual address : ffffc900107d5000 and Physical address : 4100107d5000
[ 419.263946] Form software walk: Structure Virtual address : ffffc900107d5000 and Physical address : 13f5a6000
```