CS 746 : Linux Kernel Programming

Assignment 1: Part 2
The k and v of allocating memory in the kernel

Submitted by

Aurobindo Mondal
143050082

Under the guidance of

Prof. Purushottam Kulkarni

Department of Computer Science and Engineering
IIT Bombay

February 19, 2016

kmalloc() and vmalloc()

Algorithm for kmalloc() :

- Create a structure and allocate memory using kmalloc & initialising it with dummy string

```
struct node {
      char buffer[256];
};
struct node *var;
var = kmalloc(8192, GFP_KERNEL);
strcpy(var->buffer,"Hey! this is kmalloc()");
```

- Store the virtual address of the struct var :

```
virtAddr = (unsigned long)var;
```

- Store the physical address of the struct var using macro :

```
paMacro = virt_to_phys(virtAddr);
```

- Get the physical address using page walk:
  - ✓ get pte and level
  - ✓ switch on level to find Physical Address
- Print the Virtual Address and Physical Address (using both macro and pagewalk)
- Check whether both physical address are same :

```
if(paMacro == paPagewalk)
      printk(KERN_INFO "Success!! The physical addresses are same.\n");
else
      printk(KERN_INFO "Failure!! The addresses are different.\n");
```

- Free the memory allocated using : `kfree(var)`

Algorithm for vmalloc() :

- Everything is similar to kmalloc except the syntax of vmalloc :

```
var = vmalloc(8192);
```

- Free the memory allocated using : `vfree(var)`

Pseudo Code for Page table Walk :

```
FUNCTION do_page_walk(mm_struct *mm, page, *level) :
      *level = PG_LEVEL_NONE;
      pgd = pgd_offset from mm and page;
      if(invalid pgd)
            return NULL;
      pud = pud_offset from pgd and page;
      if(invalid pud)
            return NULL;
      *level = PG_LEVEL_1G;
      if(pud_large(*pud) || !pud_present(*pud))   //1GB page
            return pud;
      pmd = pmd_offset from pud and page;
      if(invalid pmd)
            return NULL;
      *level = PG_LEVEL_2M;
      if(pmd_large(*pmd) || !pmd_present(*pmd))   //2MB page
            return pmd;
      *level = PG_LEVEL_4K;                       //4KB page
      pte = pte_offset from pmd and page;
      return pte;
```

Pseudo Code to get physical address

We get the level as well as the pte entry for the virtual address from pagewalk algo. Now we find the physical address by finding a proper offset and frame number corresponding to the pte.

```
switch(level)    {
           case PG_LEVEL_1G :
                 phyAddr = pud_pfn(*(pud_t *)pte)<<PAGE_SHIFT;
                 pOffset = virtAddr & ~PUD_PAGE_MASK;
           case PG_LEVEL_2M :
                 phyAddr = pmd_pfn(*(pmd_t *)pte)<<PAGE_SHIFT;
                 pOffset = virtAddr & ~PMD_PAGE_MASK;
      default :
                 phyAddr = pte_pfn(*pte)<<PAGE_SHIFT;
                 pOffset = virtAddr & ~PAGE_MASK;
      }
```

```
paPagewalk = ((phys_addr_t)(phyAddr | pOffset ));
```

Results and Observations :

```
$ sudo insmod kmalloc.ko
$ dmesg
[ 9084.284291] Hey! this is kmalloc()
[ 9084.284302] Virtual Address : ffff88008d96a000
[ 9084.284305] Physical Address (using Macro)  : 8d96a000
[ 9084.284308] Physical Address (using Pagewalk) : 8d96a000
[ 9084.284310] Success!! Both the physical addresses are same.

$ sudo insmod kmalloc.ko
$ dmesg
[ 9204.576359] Hey! this is vmalloc()
[ 9204.576369] Virtual Address : ffffc900106be000
[ 9204.576372] Physical Address (using Macro)  : 4100106be000
[ 9204.576375] Physical Address (using Pagewalk) : 61eef000
[ 9204.576377] Failure!! Both the physical addresses are different.
```

Try to read struct created in the previous module from another module :

```
$ sudo insmod check.ko va=0xffff88008d96a000
$ dmesg

[10214.747890] Hey! this is kmalloc()
[10214.747899] Virtual Address : ffff88008d96a000
[10214.747902] Physical Address (using Macro)  : 8d96a000
[10214.747904] Physical Address (using Pagewalk) : 8d96a000


$ sudo rmmod check && sudo insmod check.ko va=0xffffc900106be000
$ dmesg
[10304.443921] Checking done!!!
[10321.024108] Hey! this is vmalloc()
[10321.024115] Virtual Address : ffffc900106be000
[10321.024119] Physical Address (using Macro)  : 4100106be000
[10321.024121] Physical Address (using Pagewalk) : 61eef000
```

Conclusion :

- For kmalloc(), paMacro and paPagewalk are same but for vmalloc(), paMacro and paPagewalk are different;
- Check module is able to read from the struct created by the previous modules.