Mini project 3 report: Memory virtualization #2

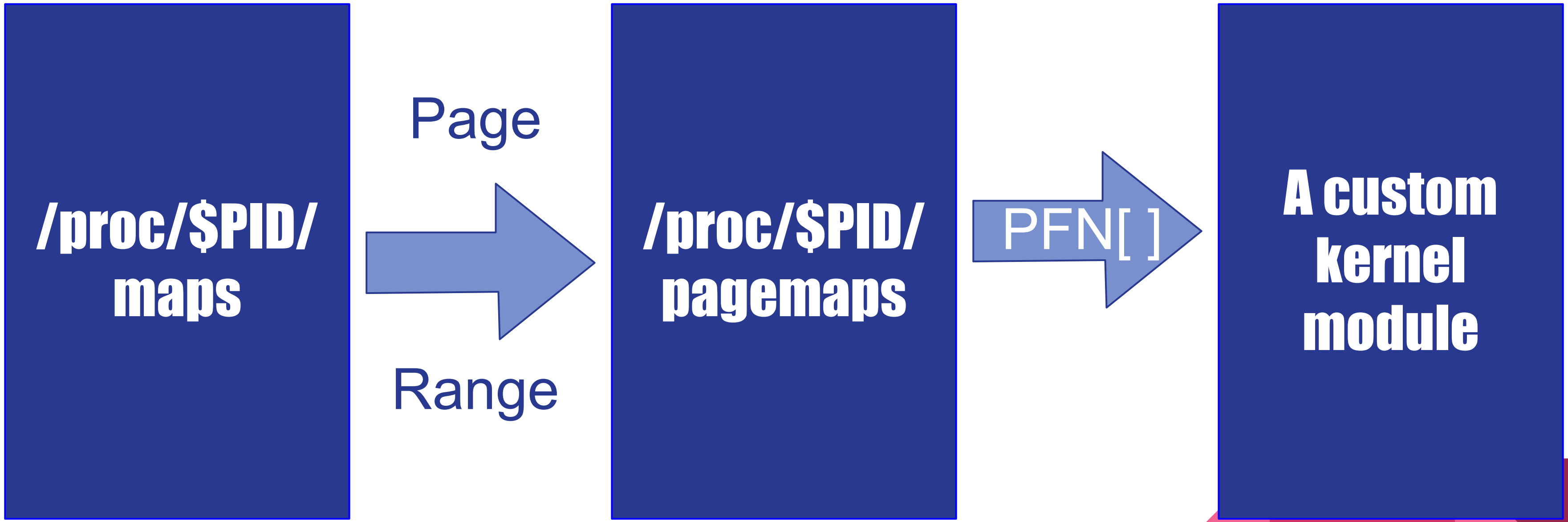
Titsanu	Wantaratorn	57070501015
Pongtorn	Siripraparwan	57070501028
Wiroat	Saeheng	57070501039
Chutichai	Tuntithawatchaikul	57070501066
Rachasak	Ragkamnerd	57070501075

Overview

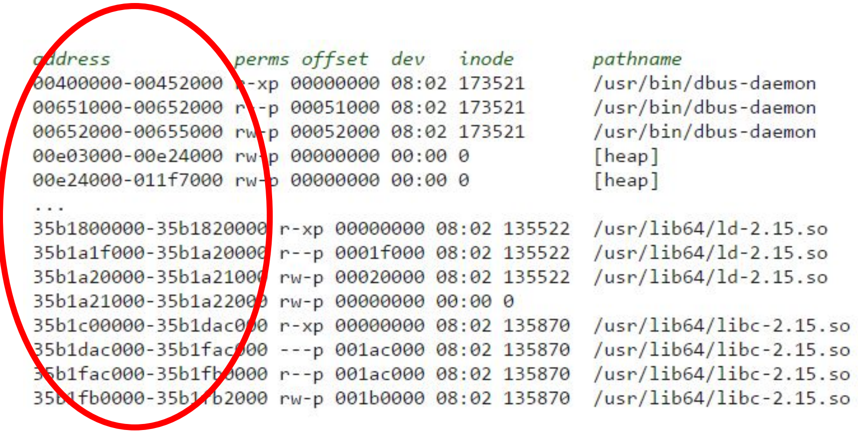
- **How it work**
- **What we have done**
  - Reading **/proc/$PID/maps**
  - Calculating and gathering **PFN**
  - Gathering **Page flag**
- **Problem**
- **Demo**
- **Q&A**

How it work

What we have done



/proc/$PID/maps



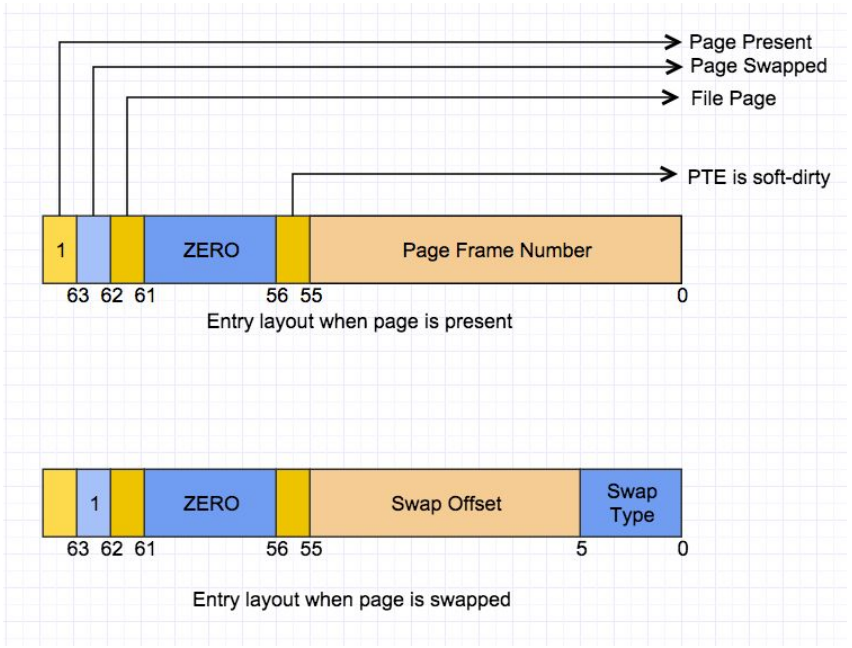
```
address           perms offset  dev   inode       pathname
00400000-00452000 r-xp 00000000 08:02 173521      /usr/bin/dbus-daemon
00651000-00652000 r--p 00051000 08:02 173521      /usr/bin/dbus-daemon
00652000-00655000 rw-p 00052000 08:02 173521      /usr/bin/dbus-daemon
00e03000-00e24000 rw-p 00000000 00:00 0           [heap]
00e24000-011f7000 rw-p 00000000 00:00 0           [heap]
...
35b1800000-35b1820000 r-xp 00000000 08:02 135522  /usr/lib64/ld-2.15.so
35b1a1f000-35b1a20000 r--p 0001f000 08:02 135522  /usr/lib64/ld-2.15.so
35b1a20000-35b1a21000 rw-p 00020000 08:02 135522  /usr/lib64/ld-2.15.so
35b1a21000-35b1a22000 rw-p 00000000 00:00 0
35b1c00000-35b1dac000 r-xp 00000000 08:02 135870  /usr/lib64/libc-2.15.so
35b1dac000-35b1fac000 ---p 001ac000 08:02 135870  /usr/lib64/libc-2.15.so
35b1fac000-35b1fb0000 r--p 001ac000 08:02 135870  /usr/lib64/libc-2.15.so
35b1fb0000-35b1fb2000 rw-p 001b0000 08:02 135870  /usr/lib64/libc-2.15.so
```

- The currently mapped memory regions and their access permissions.
- The addresses shown in this file are <u>Virtual memory address range</u>

- We'll use all of VMAR to find pages status in **/proc/$PID/pagemaps**

Read more: man proc

/proc/$PID/pagemaps

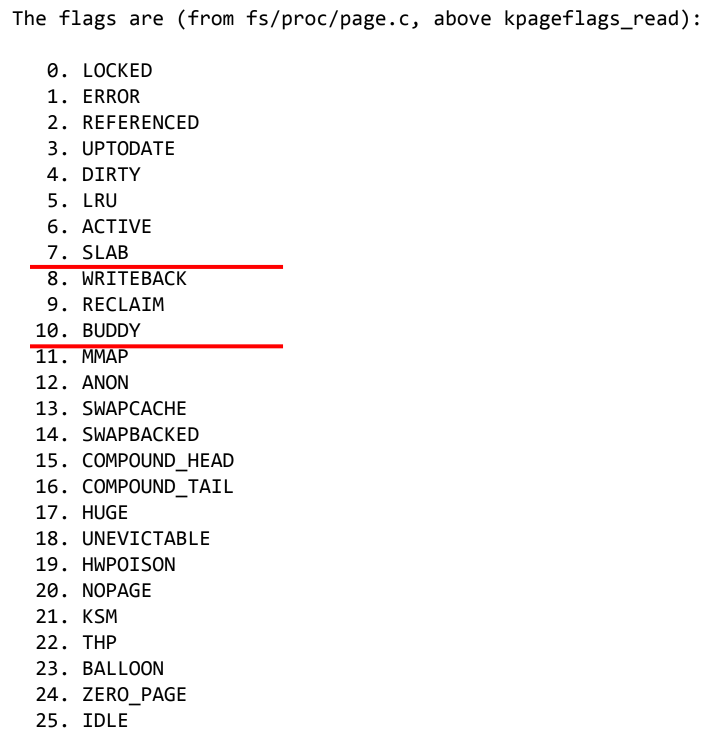


- Arrays of the page status. Each contained in 64-bit structure
- If the page is in the physical memory, the “Page Present” bit is 1 and it will contain **Page Frame Number (PFN)**
- At first, We’ll use these PFN to determine page flag in **/proc/kpageflag**
- But now we must use kernel mode...

Read more: man proc

Figure from http://blog.jeffli.me/blog/2014/11/08/pagemap-interface-of-linux-explained/

/proc/kpageflags



```
The flags are (from fs/proc/page.c, above kpageflags_read):

     0. LOCKED
     1. ERROR
     2. REFERENCED
     3. UPTODATE
     4. DIRTY
     5. LRU
     6. ACTIVE
     7. SLAB
     8. WRITEBACK
     9. RECLAIM
    10. BUDDY
    11. MMAP
    12. ANON
    13. SWAPCACHE
    14. SWAPBACKED
    15. COMPOUND_HEAD
    16. COMPOUND_TAIL
    17. HUGE
    18. UNEVICTABLE
    19. HWPOISON
    20. NOPAGE
    21. KSM
    22. THP
    23. BALLOON
    24. ZERO_PAGE
    25. IDLE
```

For SLAB and BUDDY allocation, If the bit is 1, the method is used to allocate the memory

Read more: man proc

Figure from http://lxr.free-electrons.com/source/tools/vm/page-types.c

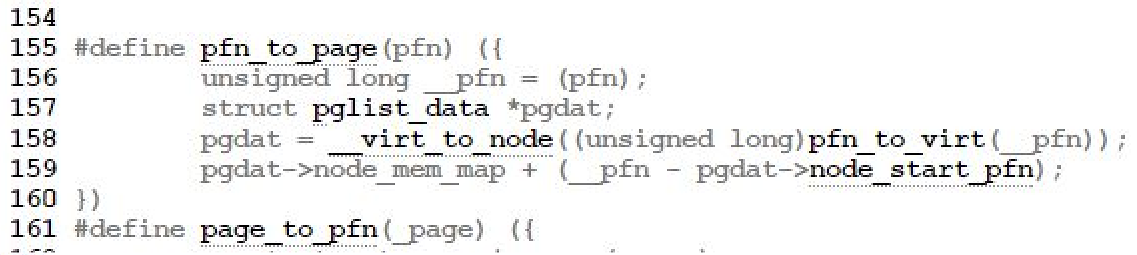
Memory Allocation

Direct Memory Access (DMA) Pages - less than 16 MB
Normal addressable pages (NormalMem) - 16-896 MB
Dynamically mapped pages (HighMem) - more than 896 MB

Gathering Page Flag

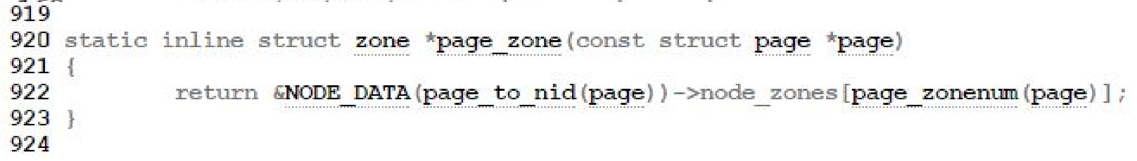
In **kernel mode**, we can use sets of the following function, and structure element to get data:

1. pfn_to_page -- convert PFN to physical memory page detail.
2. page_zone -- get detail of memory zone used in the page.
3. (Optional) page_zonenum -- get the zone index used in the page
4. page->flags -- get flags contained in the page



```

#define pfn_to_page(pfn) ({
        unsigned long __pfn = (pfn);
        struct pglist_data *pgdat;
        pgdat = __virt_to_node((unsigned long)pfn_to_virt(__pfn));
        pgdat->node_mem_map + (__pfn - pgdat->node_start_pfn);
})
#define page_to_pfn(_page) ({
```

page_mm.h



```

static inline struct zone *page_zone(const struct page *page)
{
        return &NODE_DATA(page_to_nid(page))->node_zones[page_zonenum(page)];
}

```

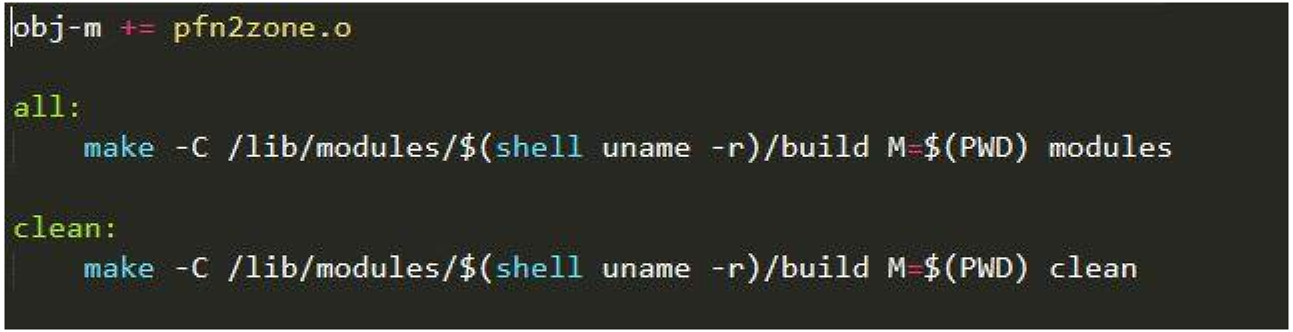
mm.h

Kernel Module

Kernel module is a new way to compile kernel by building process for external loadable modules that has been integrated into standard kernel build mechanism.

What we need to compile a basic kernel module :

- A Makefile to compile a module

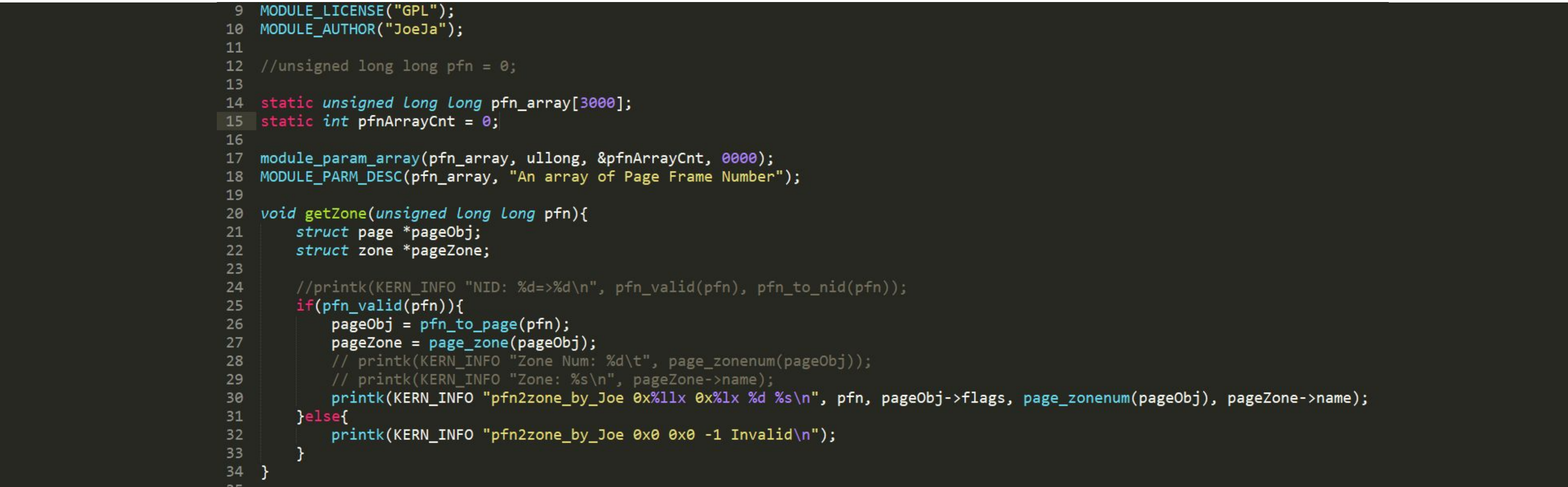


```
obj-m += pfn2zone.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Inside of a Makefile file

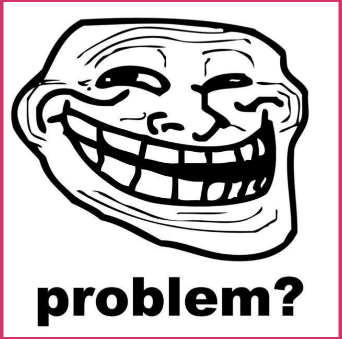
Kernel Module

- In this case, we have a .C module file named pfn2zone.c
- In terminal, use command "make" to compile it.
- Then, use "insmod ./pfn2zone.ko [parameters]" to insert to kernel
- In case of there's a kernel module that we don't want it anymore use "rmmod modulename" to remove it from your kernel.

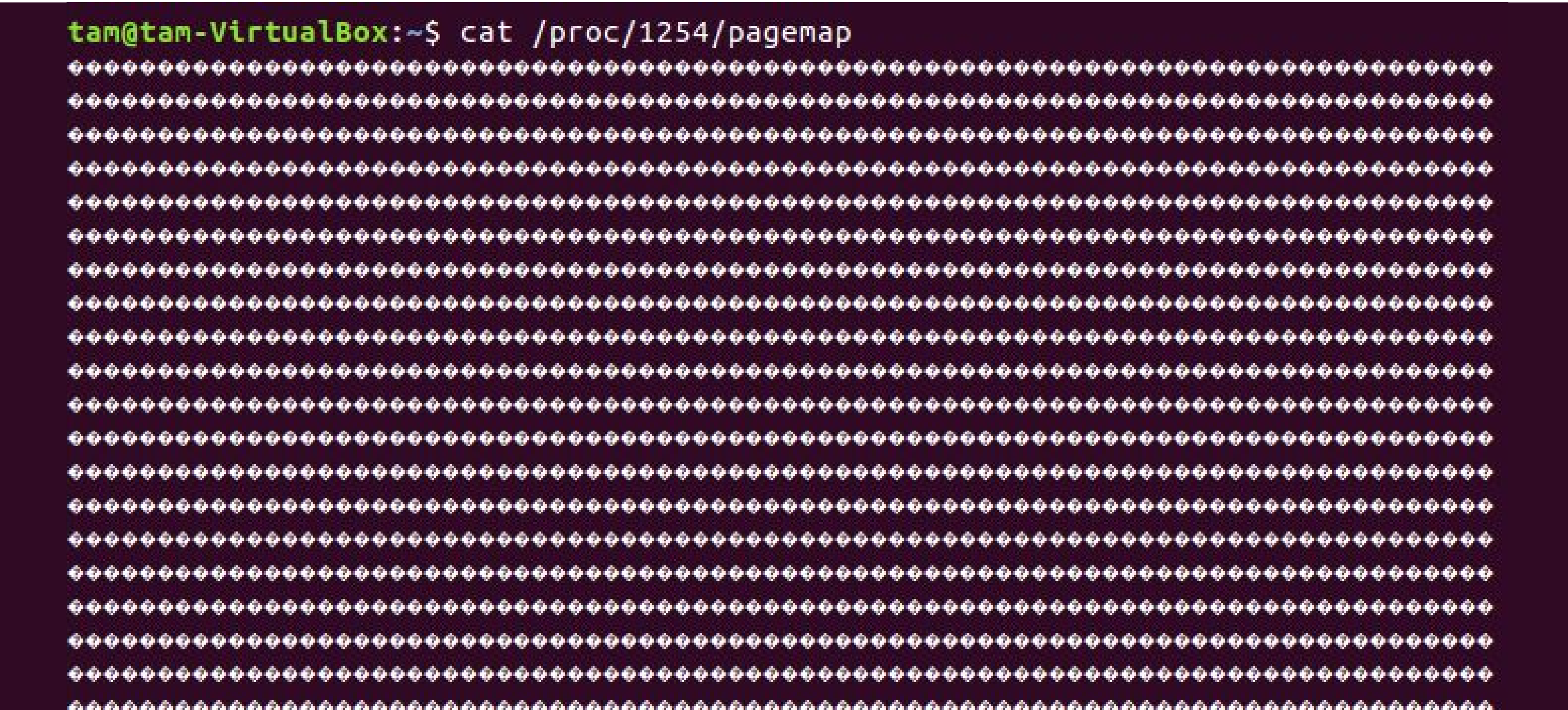


```c
MODULE_LICENSE("GPL");
MODULE_AUTHOR("JoeJa");

//unsigned long long pfn = 0;

static unsigned long long pfn_array[3000];
static int pfnArrayCnt = 0;

module_param_array(pfn_array, ullong, &pfnArrayCnt, 0000);
MODULE_PARM_DESC(pfn_array, "An array of Page Frame Number");

void getZone(unsigned long long pfn){
    struct page *pageObj;
    struct zone *pageZone;

    //printk(KERN_INFO "NID: %d=>%d\n", pfn_valid(pfn), pfn_to_nid(pfn));
    if(pfn_valid(pfn)){
        pageObj = pfn_to_page(pfn);
        pageZone = page_zone(pageObj);
        // printk(KERN_INFO "Zone Num: %d\t", page_zonenum(pageObj));
        // printk(KERN_INFO "Zone: %s\n", pageZone->name);
        printk(KERN_INFO "pfn2zone_by_Joe 0x%llx 0x%lx %d %s\n", pfn, pageObj->flags, page_zonenum(pageObj), pageZone->name);
    }else{
        printk(KERN_INFO "pfn2zone_by_Joe 0x0 0x0 -1 Invalid\n");
    }
}
```

Problem



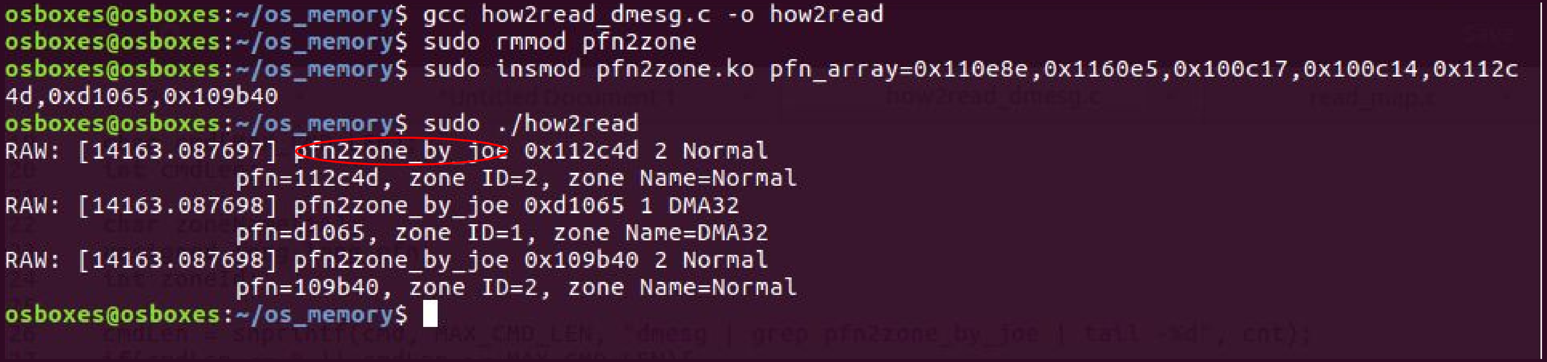
Reading /proc/$PID/pagemap



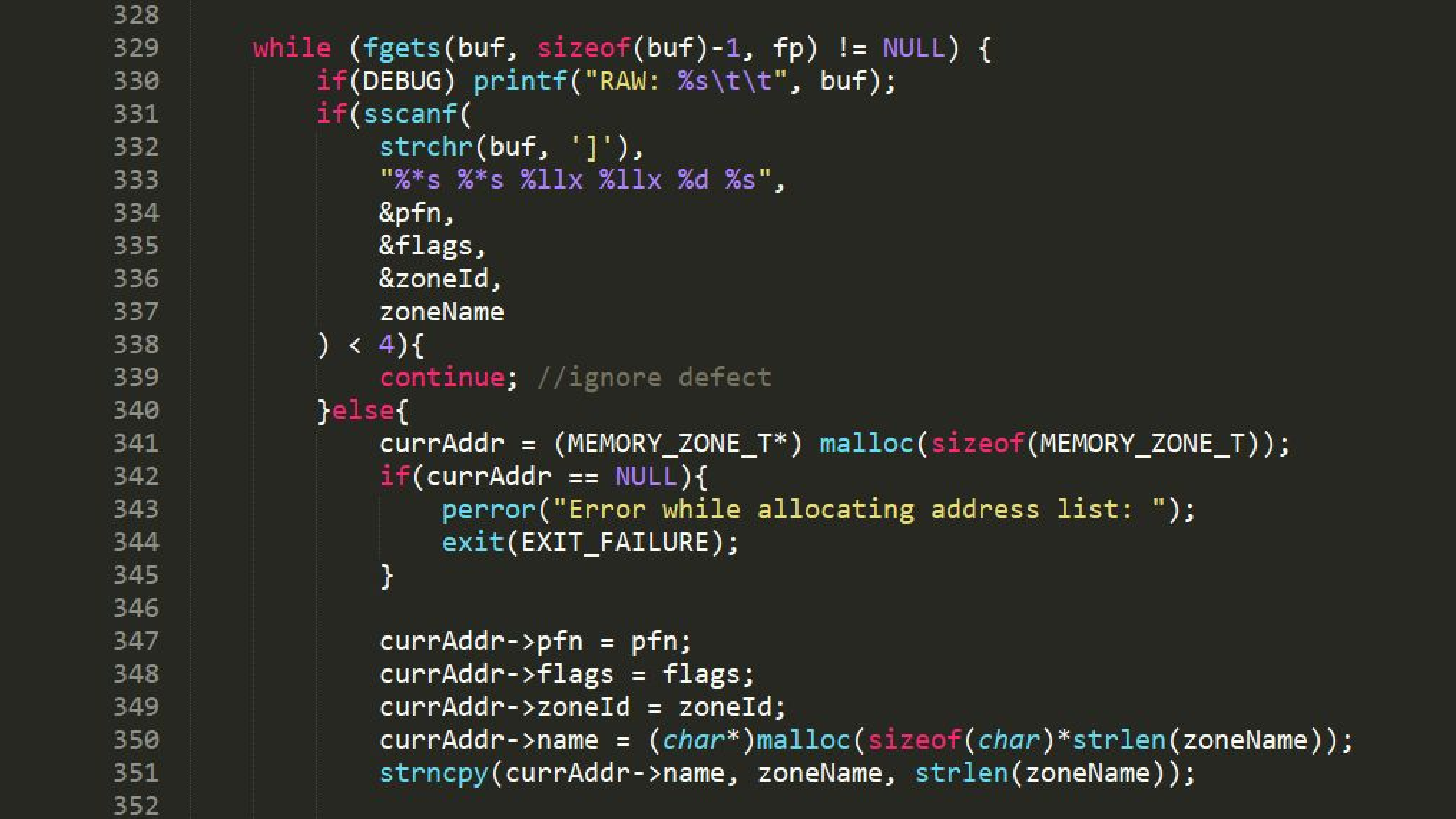
```
tam@tam-VirtualBox:~$ cat /proc/1254/pagemap
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
��������������������������������������������������������������������������������
```

Issue in the kernel mode

- Result must show output via printk -> results are in the "DMESG"
  - Solution: Create DMESG reader, add signature prefix before the real interested value. So we can use **grep** to filter out unwanted data rows from another programs.
  - Use * trick from sscanf: ignore the following format value.



```
osboxes@osboxes:~/os_memory$ gcc how2read_dmesg.c -o how2read
osboxes@osboxes:~/os_memory$ sudo rmmod pfn2zone
osboxes@osboxes:~/os_memory$ sudo insmod pfn2zone.ko pfn_array=0x110e8e,0x1160e5,0x100c17,0x100c14,0x112c
4d,0xd1065,0x109b40
osboxes@osboxes:~/os_memory$ sudo ./how2read
RAW: [14163.087697] pfn2zone_by_joe 0x112c4d 2 Normal
                pfn=112c4d, zone ID=2, zone Name=Normal
RAW: [14163.087698] pfn2zone_by_joe 0xd1065 1 DMA32
                pfn=d1065, zone ID=1, zone Name=DMA32
RAW: [14163.087698] pfn2zone_by_joe 0x109b40 2 Normal
                pfn=109b40, zone ID=2, zone Name=Normal
osboxes@osboxes:~/os_memory$ 
```



```c

    while (fgets(buf, sizeof(buf)-1, fp) != NULL) {
        if(DEBUG) printf("RAW: %s\t\t", buf);
        if(sscanf(
            strchr(buf, ']'),
            "%*s %*s %llx %llx %d %s",
            &pfn,
            &flags,
            &zoneId,
            zoneName
        ) < 4){
            continue; //ignore defect
        }else{
            currAddr = (MEMORY_ZONE_T*) malloc(sizeof(MEMORY_ZONE_T));
            if(currAddr == NULL){
                perror("Error while allocating address list: ");
                exit(EXIT_FAILURE);
            }

            currAddr->pfn = pfn;
            currAddr->flags = flags;
            currAddr->zoneId = zoneId;
            currAddr->name = (char*)malloc(sizeof(char)*strlen(zoneName));
            strncpy(currAddr->name, zoneName, strlen(zoneName));

```

So
Kernel
⋮

Demo

Q&A

Reference

- https://fivelinesofcode.blogspot.com/2014/03/how-to-translate-virtual-to-physical.html
- http://lxr.free-electrons.com/source/tools/vm/page-types.c
- http://www.tldp.org/LDP/lkmpg/2.6/html/lkmpg.html
- https://www.gnu.org/software/libc/manual/html_node/Permission-Bits.html
- https://stackoverflow.com/questions/5947286/how-can-linux-kernel-modules-be-loaded-from-c-code
- http://www.cplusplus.com/reference/cstdio/scanf/