SSE3052: Embedded Systems Practice

Jinkyu Jeong
jinkyu@skku.edu
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

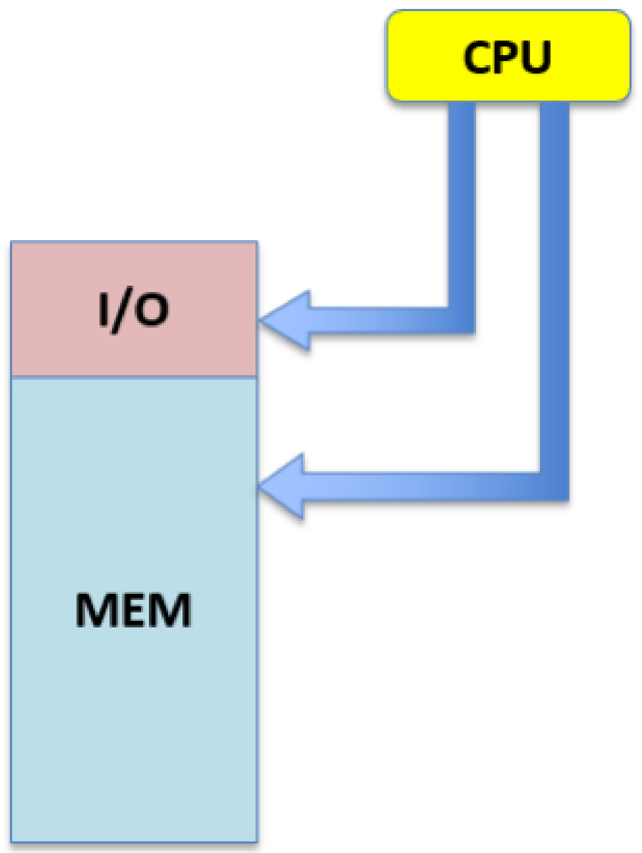
Overview

- Last week, we attached new virtual devices and write the data to the "memory-mapped area" via <u>new</u> *system call*
- For this week, we will read and write the data via "*mmap*" and "*ioctl*"

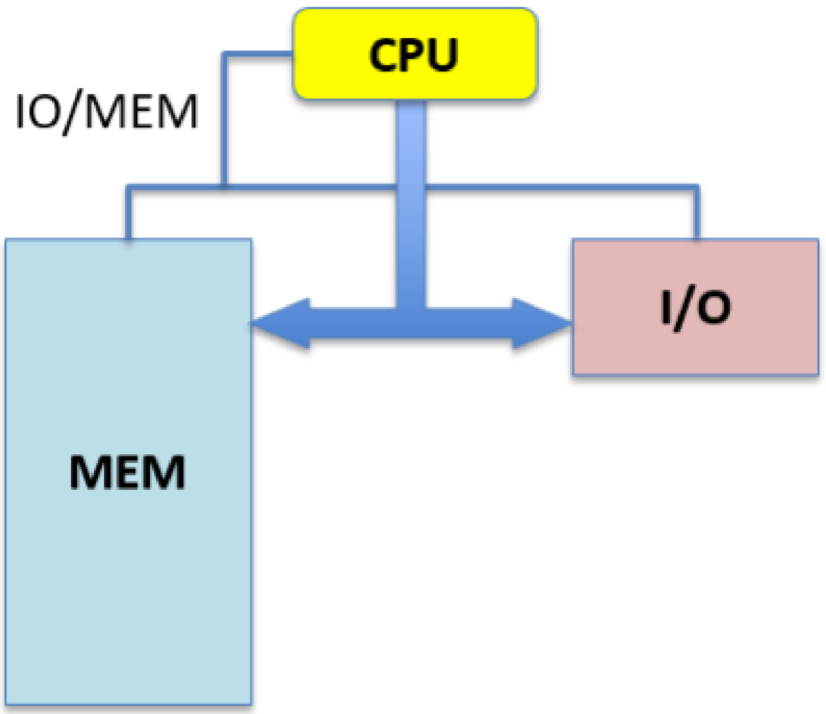
- Keywords: memory-mapped area, mmap, ioctl

Memory & Port Mapped I/O

Memory mapped I/O



Port mapped I/O



Memory & Port Mapped I/O

- Memory mapped I/O
  - Device & memory share same address space
  - I/O looks just like memory load/store
  - No special command for I/O
- Port mapped I/O
  - Separated address space
  - Need I/O or memory select lines
  - Special command for I/O
    - IN / OUT

Two Importants

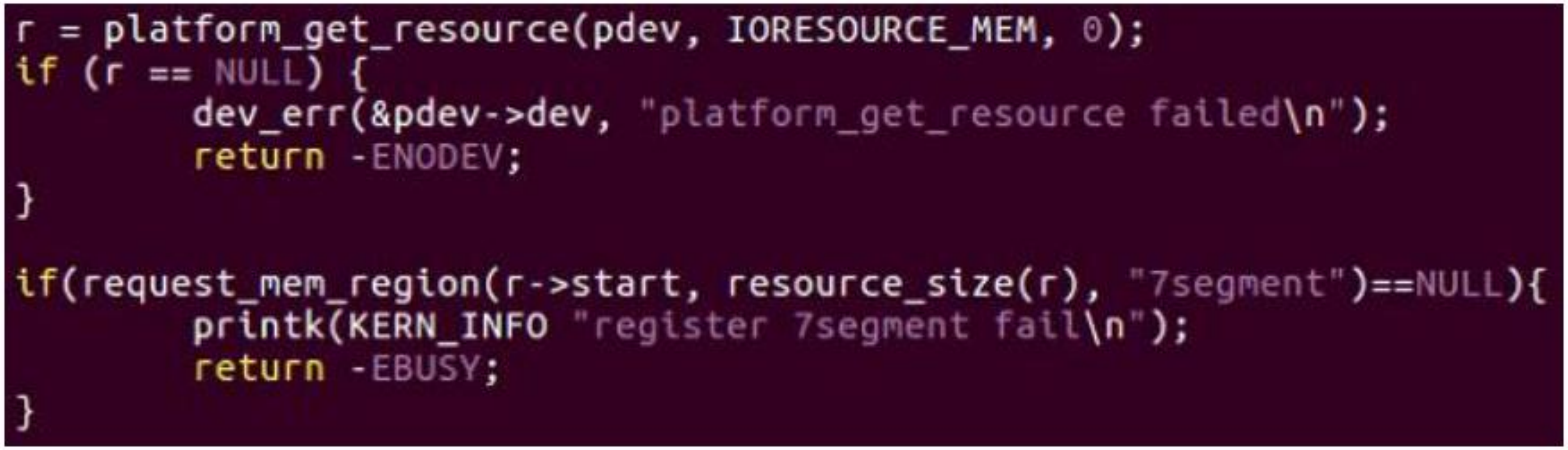
- Base address of the device's mapped area
  - What is the 7 segment's base(physical) address?
  - What is the LED's base(physical) address?
- Size of the device's mapped area
  - What is the 7 segment's I/O device size?
  - What is the LED I/O devices size?

I/O Memory Region Allocation API

- struct resource *request_mem_region(unsigned long start, unsigned long len, char *name)
- void release_mem_region(unsigned long start, unsigned long len)
- int check_mem_region(unsigned long start, unsigned long len)

Register Memory Region

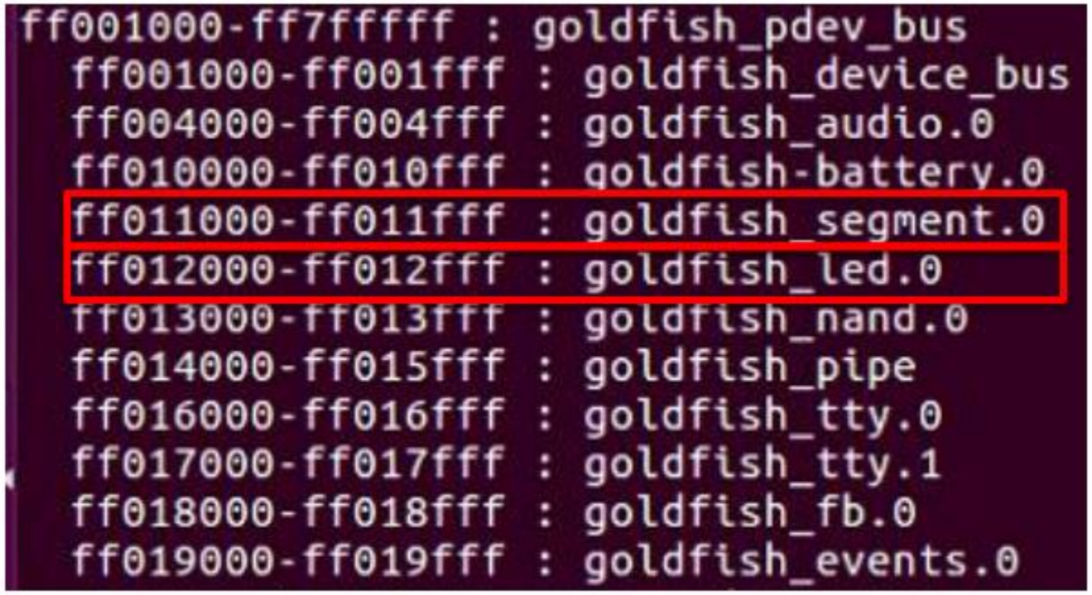
- Add these codes in the device driver
  - drivers/misc/goldfish_segment.c
  - drivers/misc/goldfish_led.c



```
r = platform_get_resource(pdev, IORESOURCE_MEM, 0);
if (r == NULL) {
        dev_err(&pdev->dev, "platform_get_resource failed\n");
        return -ENODEV;
}

if(request_mem_region(r->start, resource_size(r), "7segment")==NULL){
        printk(KERN_INFO "register 7segment fail\n");
        return -EBUSY;
}
```

Check Device's Information

- $adb shell
- $su
- $cat /proc/iomem
  - We can figure out base address & size



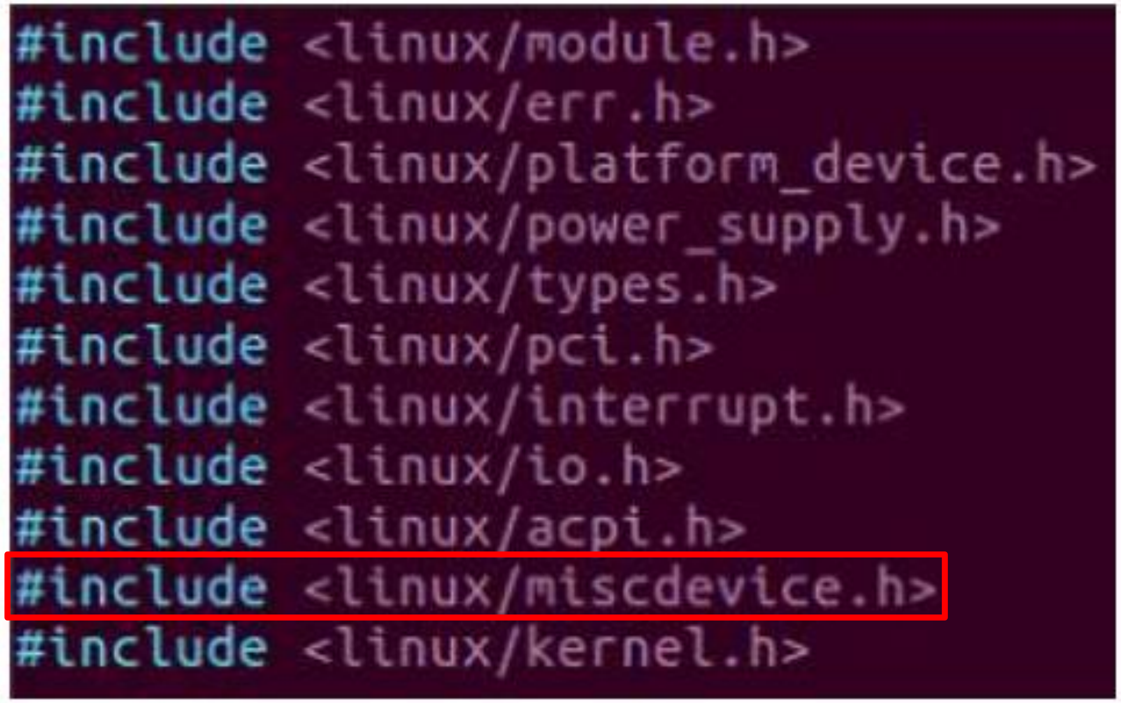
```
ff001000-ff7fffff : goldfish_pdev_bus
  ff001000-ff001fff : goldfish_device_bus
  ff004000-ff004fff : goldfish_audio.0
  ff010000-ff010fff : goldfish-battery.0
  ff011000-ff011fff : goldfish_segment.0
  ff012000-ff012fff : goldfish_led.0
  ff013000-ff013fff : goldfish_nand.0
  ff014000-ff015fff : goldfish_pipe
  ff016000-ff016fff : goldfish_tty.0
  ff017000-ff017fff : goldfish_tty.1
  ff018000-ff018fff : goldfish_fb.0
  ff019000-ff019fff : goldfish_events.0
```

Check the Device File

- $adb shell
- $ls /dev/
  - there are no segment & led device file's
  - we should register those devices

Register misc_device

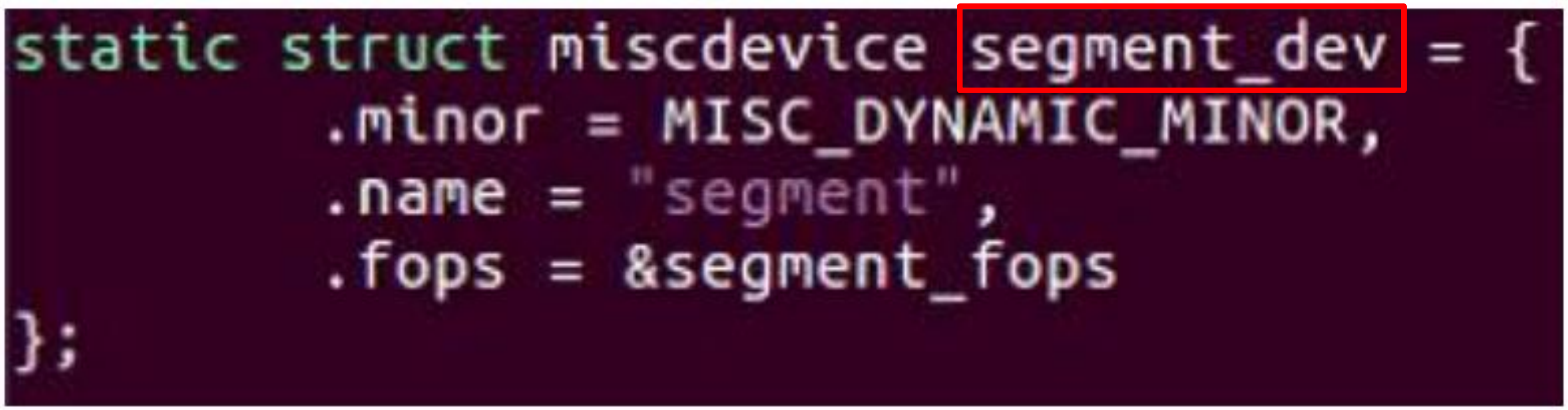
- 1. Make a misc device
- 2. Make a file operations for your device
- 3. Include miscdevice.h in device code



```
#include <linux/module.h>
#include <linux/err.h>
#include <linux/platform_device.h>
#include <linux/power_supply.h>
#include <linux/types.h>
#include <linux/pci.h>
#include <linux/interrupt.h>
#include <linux/io.h>
#include <linux/acpi.h>
#include <linux/miscdevice.h>
#include <linux/kernel.h>
```

Make a misc_device (1)

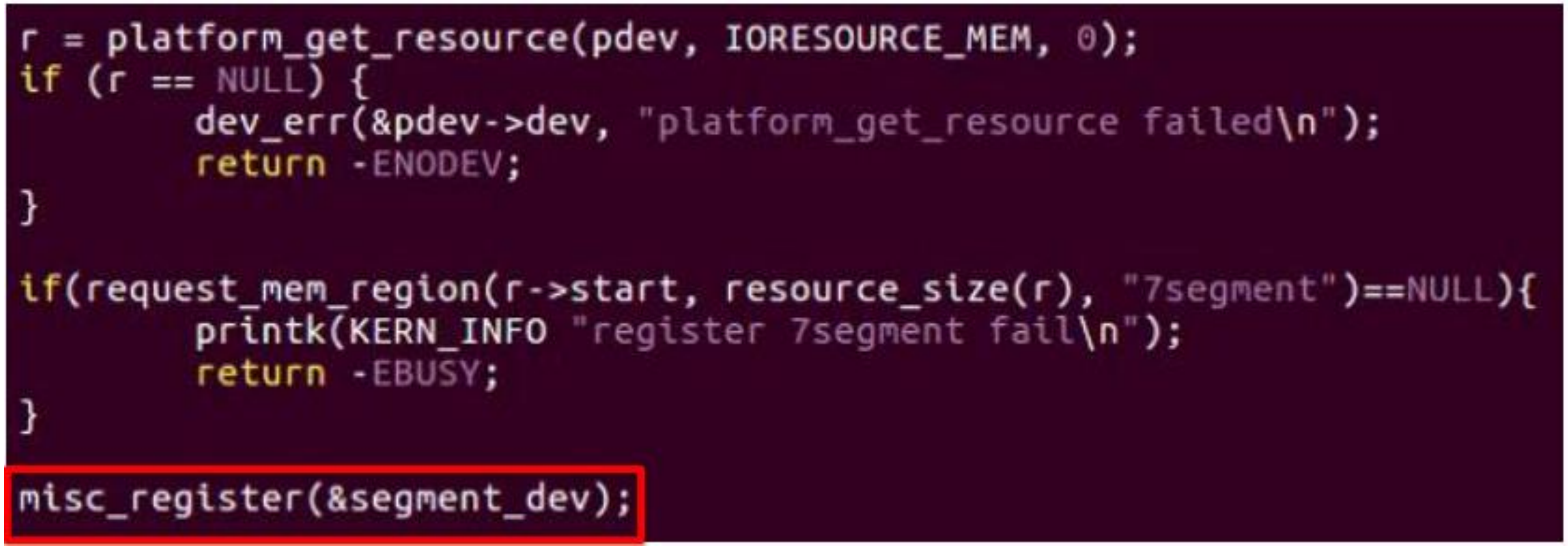
- Declare a misc device



```
static struct miscdevice segment_dev = {
        .minor = MISC_DYNAMIC_MINOR,
        .name = "segment",
        .fops = &segment_fops
};
```

Make a misc_device (2)

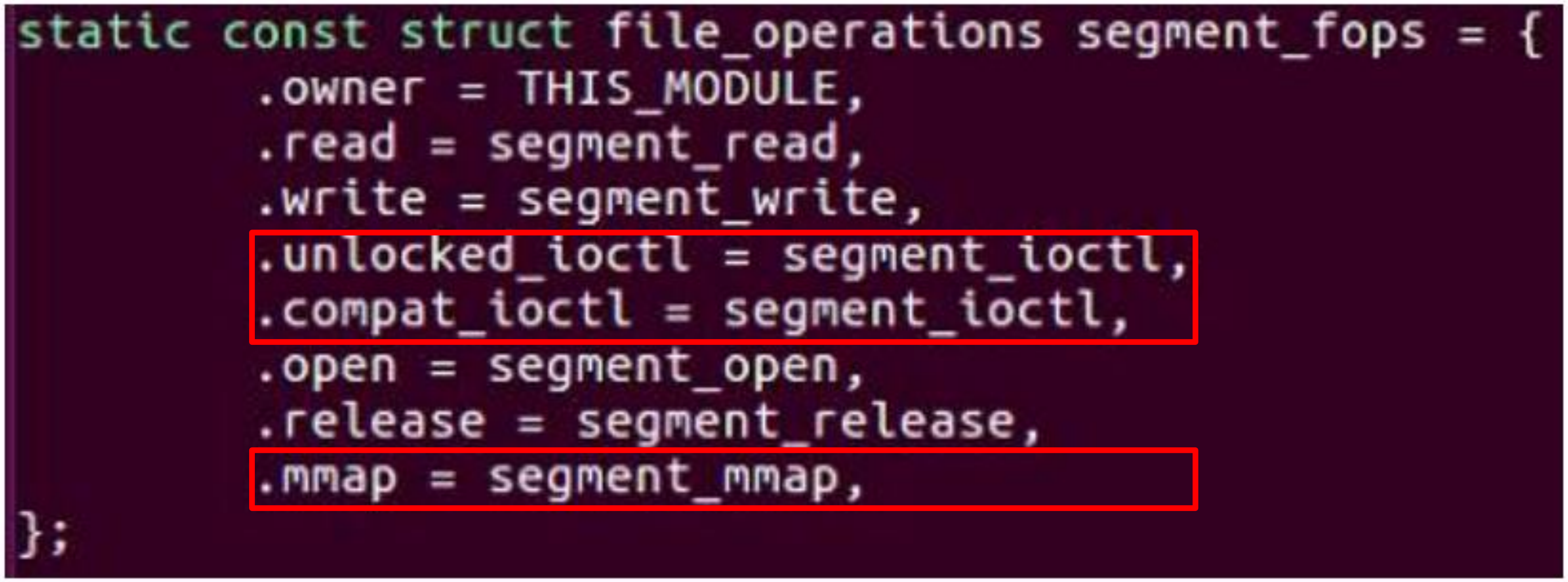
- Register a misc device (goldfish_segment.c)



```
r = platform_get_resource(pdev, IORESOURCE_MEM, 0);
if (r == NULL) {
        dev_err(&pdev->dev, "platform_get_resource failed\n");
        return -ENODEV;
}

if(request_mem_region(r->start, resource_size(r), "7segment")==NULL){
        printk(KERN_INFO "register 7segment fail\n");
        return -EBUSY;
}

misc_register(&segment_dev);
```

Make File Operations (1)

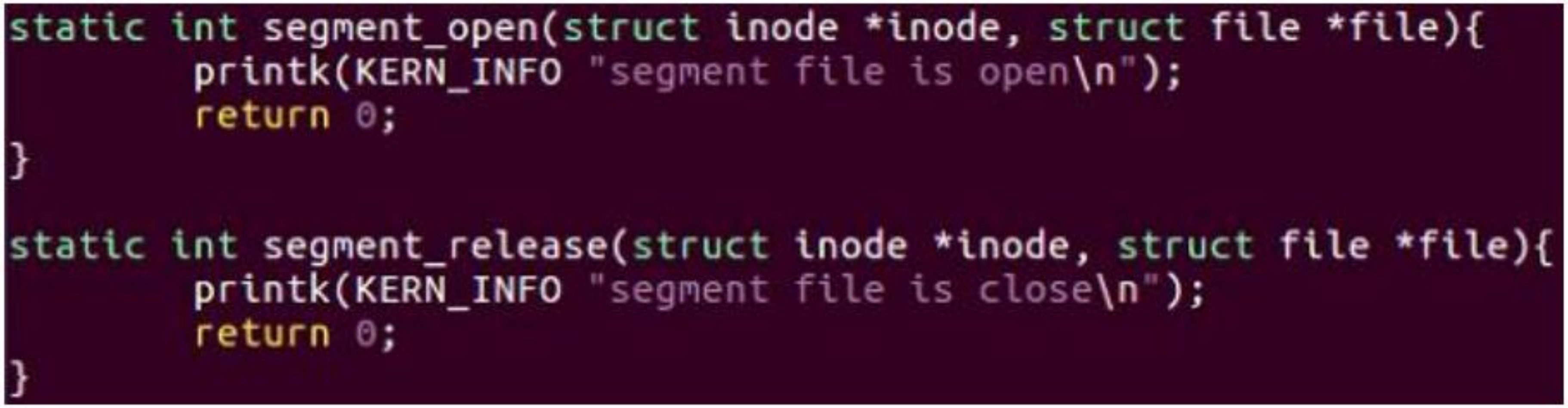
- Make a structure of file operations for misc device



```
static const struct file_operations segment_fops = {
        .owner = THIS_MODULE,
        .read = segment_read,
        .write = segment_write,
        .unlocked_ioctl = segment_ioctl,
        .compat_ioctl = segment_ioctl,
        .open = segment_open,
        .release = segment_release,
        .mmap = segment_mmap,
};
```

Make File Operations (2)

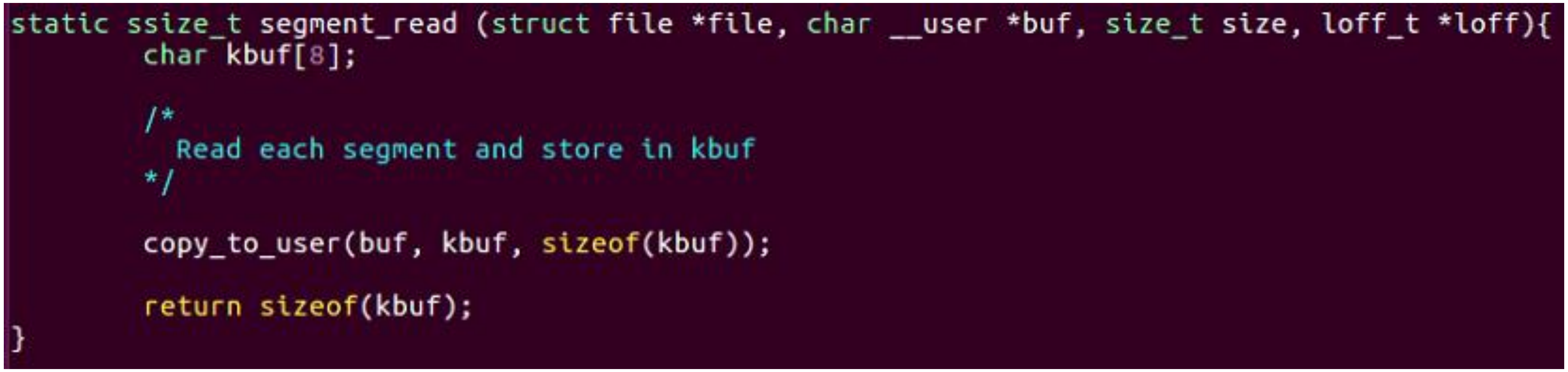
- Define each operations



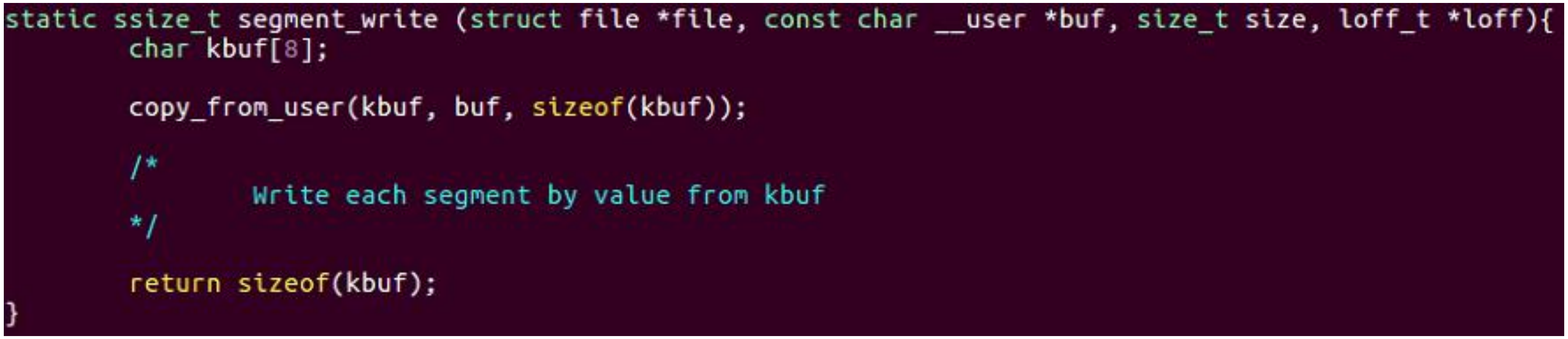
```
static int segment_open(struct inode *inode, struct file *file){
        printk(KERN_INFO "segment file is open\n");
        return 0;
}

static int segment_release(struct inode *inode, struct file *file){
        printk(KERN_INFO "segment file is close\n");
        return 0;
}
```

Make File Operations (3)

- Read & write operation (for your own)

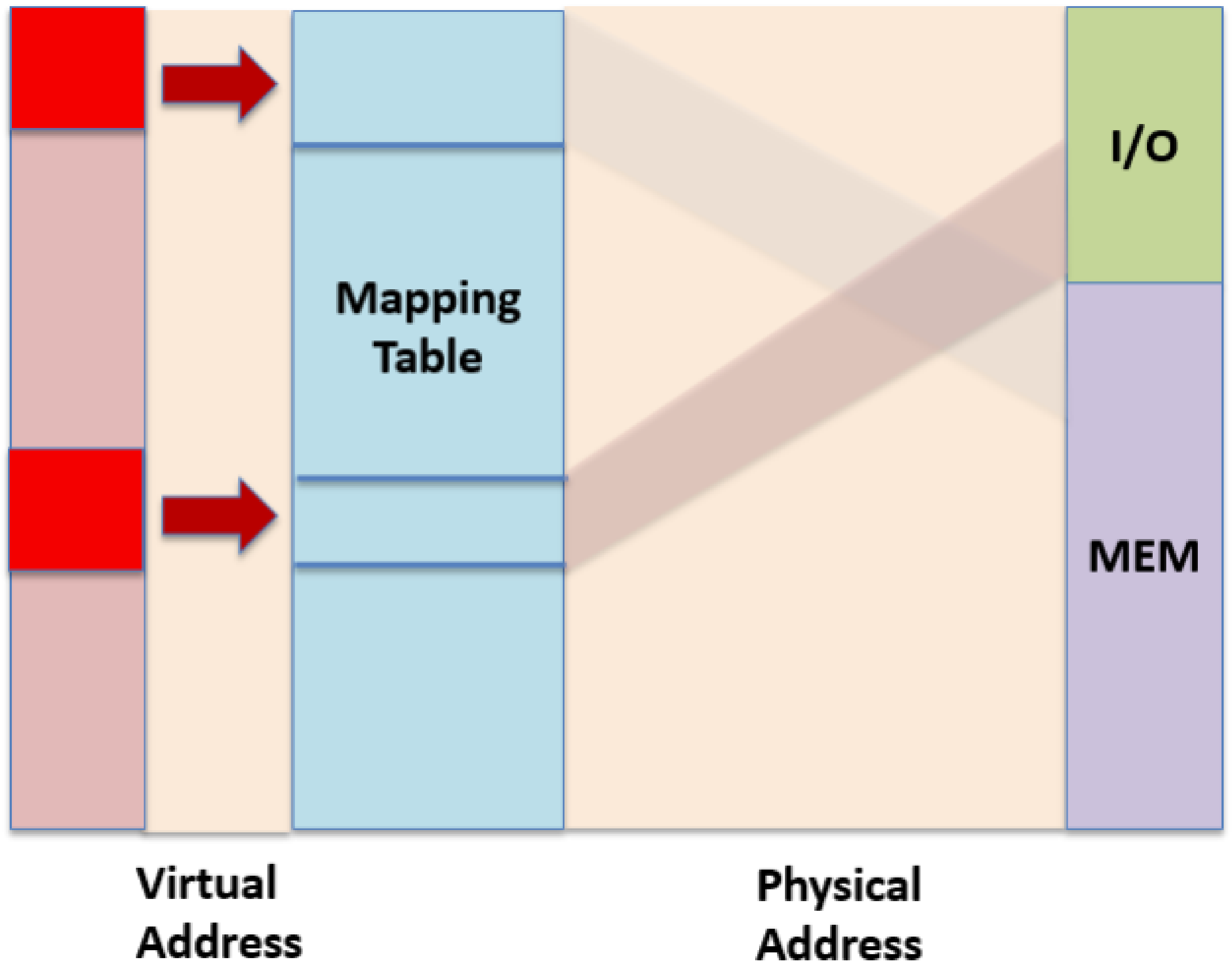


```
static ssize_t segment_read (struct file *file, char __user *buf, size_t size, loff_t *loff){
        char kbuf[8];

        /*
          Read each segment and store in kbuf
        */

        copy_to_user(buf, kbuf, sizeof(kbuf));

        return sizeof(kbuf);
}
```



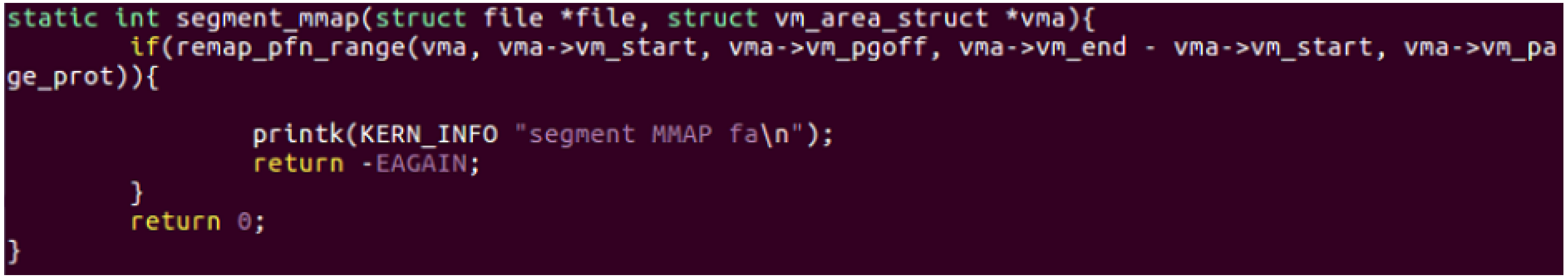
```
static ssize_t segment_write (struct file *file, const char __user *buf, size_t size, loff_t *loff){
        char kbuf[8];

        copy_from_user(kbuf, buf, sizeof(kbuf));

        /*
                Write each segment by value from kbuf
        */

        return sizeof(kbuf);
}
```

I/O via *mmap*



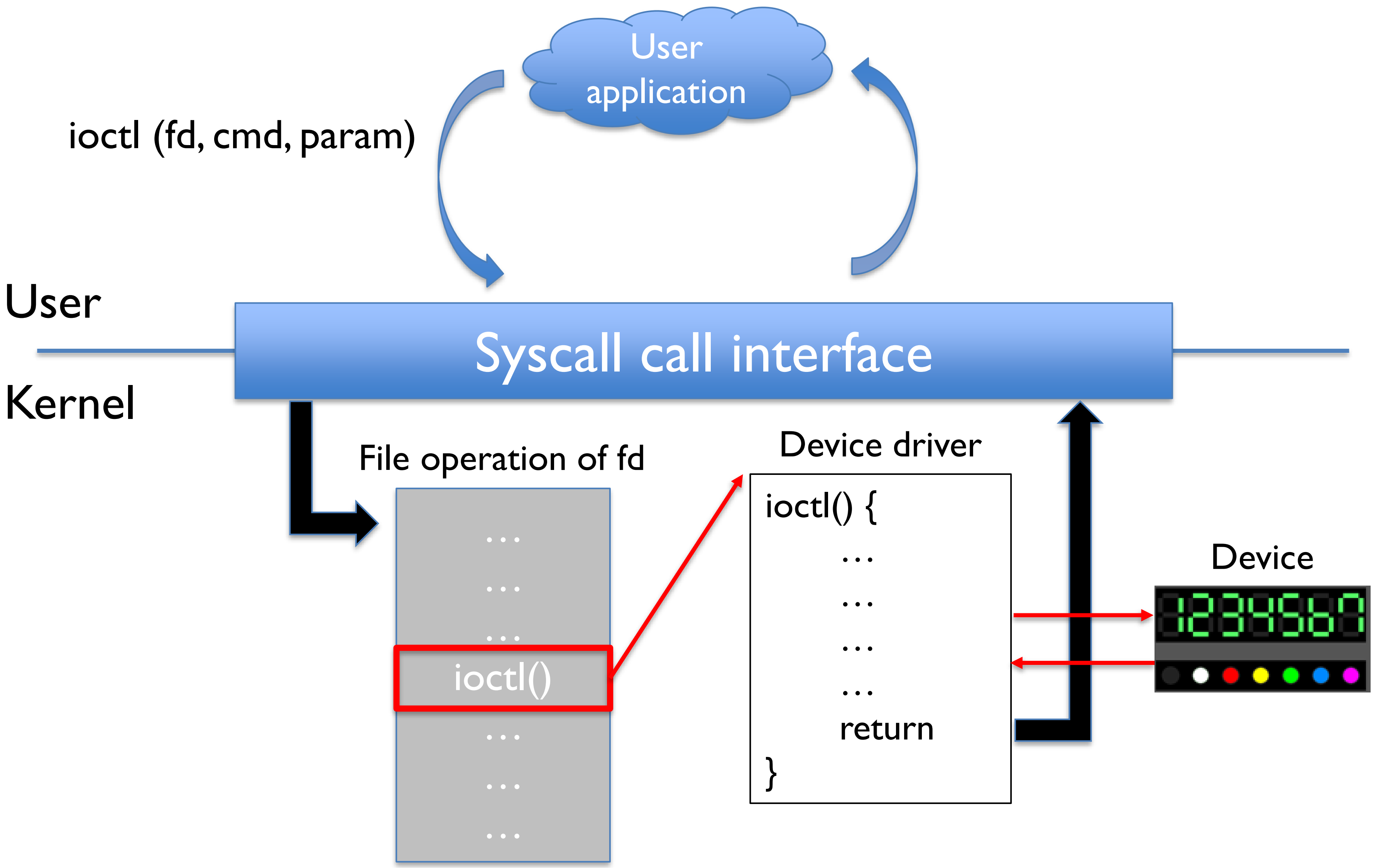
Memory Mapping API

- int remap_pfn_range(struct vm_area_struct *vma, unsigned long addr, unsigned long pfn, unsigned long size, pgprot_t prot)



```
static int segment_mmap(struct file *file, struct vm_area_struct *vma){
        if(remap_pfn_range(vma, vma->vm_start, vma->vm_pgoff, vma->vm_end - vma->vm_start, vma->vm_pa
ge_prot)){

                printk(KERN_INFO "segment MMAP fa\n");
                return -EAGAIN;
        }
        return 0;
}
```

I/O via *ioctl*



User application
ioctl (fd, cmd, param)
User
Kernel
Syscall call interface
File operation of fd
…
…
…
ioctl()
…
…
…
Device driver
ioctl() {
…
…
…
…
return
}
Device

Make File Operations - ioctl

- static long segment_ioctl(struct file *file, unsigned int cmd, unsigned long para)

- static long led_ioctl(struct file *file, unsigned int cmd, unsigned long para)

Make File Operations - ioctl

User code

```
#include <stdio.h>

#define CMD1 0x04      //print input
#define CMD2 0x05      //print (input + 1)
#define CMD3 0x06      //print (input + 2)

int main(int argc, char *argv[]){
        int fd;

        fd = open("/dev/segment", O_RDWR);

        ioctl(fd, CMD3, 3);

        close(fd);

        return 0;
}
```

Device driver

```
#define CMD1 0x04
#define CMD2 0x05
#define CMD3 0x06
…

static long segment_ioctl(struct file *file,
unsigned int cmd, unsigned long para) {
        swtich(cmd) {
                case CMD1: …;break;
                case CMD2: …;break;
                case CMD3:
                        printk("%d\n", (int)para + 1);
                        break;
        };
}

…
```

Questions?

- If you have questions,
  - please use i-Campus (토론>수업 Q&A 토론) or email
    - minwoo.ahn@csi.skku.edu
    - bumsuk.kim@csi.skku.edu