# Device Driver

# Linux Loadable Kernel Modules (LKM)

- A way dynamically ADD code to the Linux kernel
- LKM is usually used for dynamically add
  - device drivers
  - filesystem drivers
  - system calls
  - network drivers
  - executable interpreters

# Why use LKMs

- Need not to rebuild kernel
- Diagnosing system problems
  - Easier to locate in which part of the kernel problems occur
- Modules are faster to maintain and debug
- LKMs are not slower than base kernel parts
- However, if the system startup is dependent on a module, it has to be included in the base kernel
  - E.g. File system driver

# Kernel Module Programming

- Kernel module은 module_init(init_func);에서 시작
- Kernel module은 module_exit(exit_func);에서 종료

- Init_func과 exit_func은 static function.

```
static int module_begin(void) {
//TODO:…
return 0;
}
static void module_end(void) {
//TODO:…
}
module_init(module_begin);
module_exit(module_end);
```

# Module Programming Example

- Simple Module Program (hello.c)

```
/* Module example
FILE : hello_module.c */
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Huins");
static int module_begin(void) {
printk(KERN_ALERT "Hello, Wellcome to module!!\n");
return 0;
}
static void module_end(void) {
printk(KERN_ALERT "Goodbye, Exit module!!\n");
}
module_init(module_begin);
module_exit(module_end);
```

# Module Programming Example

- Makefile

```
obj-m := hello_module.o
KDIR := /work/achroimx6q/kernel
PWD := $(shell pwd)
all: driver

driver:
	$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules

clean:
	rm -rf *.ko
	rm -rf *.mod.*
	rm -rf *.o
```

- Run on the target

```
# insmod  hello_module.ko        (모듈을 커널에 적재)
# more  /proc/modules            (적재된 모듈 보기)
# rmmod  hello_module            (모듈 삭제하기)
```

# 모듈 관련 리눅스 명령어

- 모듈 명령어 요약

| 명령어 | 용 도 |
|---|---|
| insmod | module을 설치(install) |
| rmmod | 실행중인 modules을 제거(unload) |
| lsmod | Load된 module들의 정보를 표시 |
| depmod | 커널 내부에 적재된 모듈간의 의존성을 검사한다. |
| modprobe | 모듈간 의존성을 검사하여 그 결과 누락된 다른 모듈을 찾아서 적재한다. |
| modinfo | 목적화일을 검사해서 관련된 정보를 표시 |

# Device Driver Overview

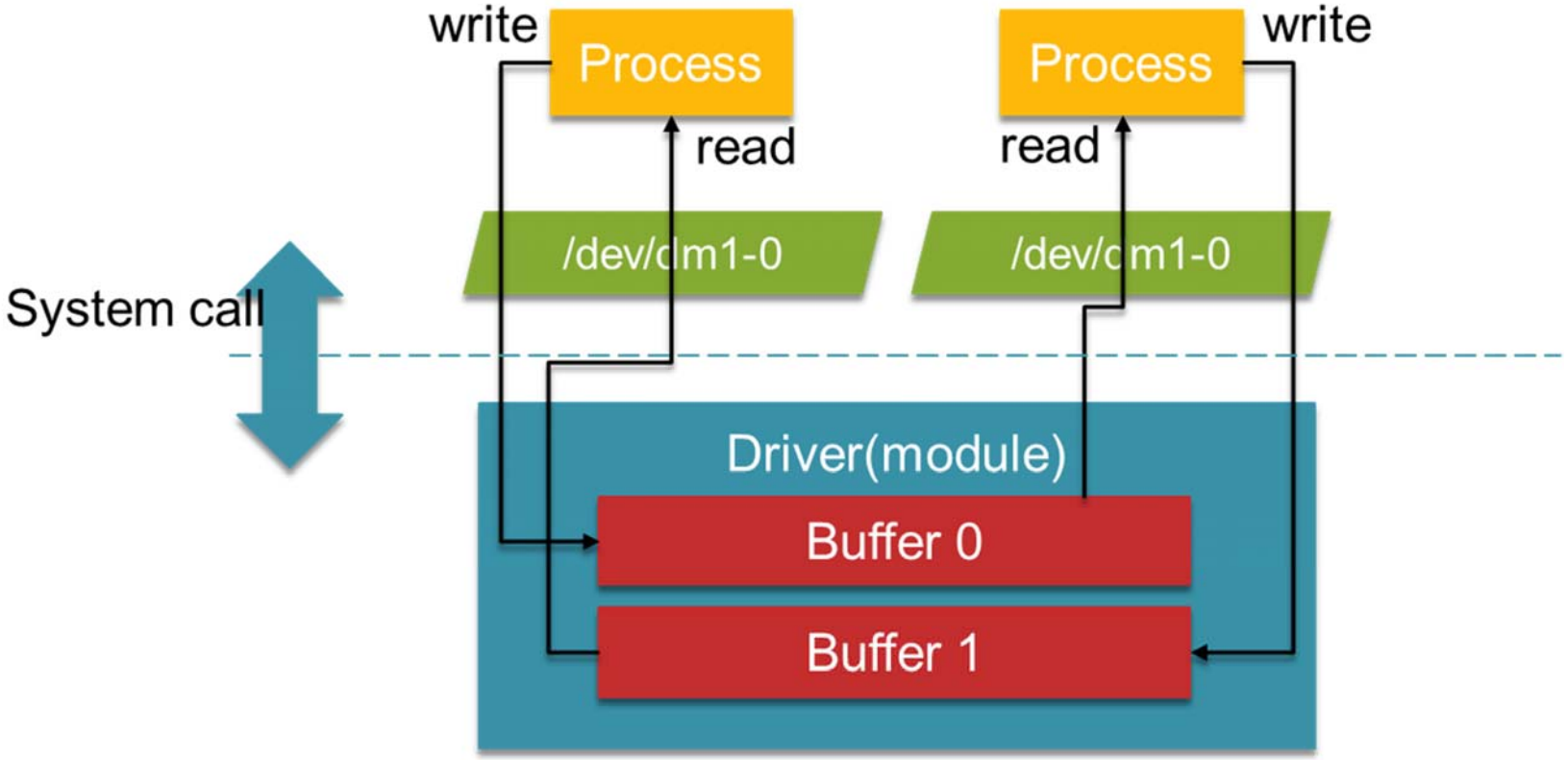
- IO device제어를 위한 device-specific code는 주로 device 제조사에 의해 만들어진다.
  - Device driver는 OS kernel의 일부분
  - OS와 함께 컴파일
  - 동적으로 OS가 실행되는 동안 load
- Categories
  - Block devices
  - Char devices

# Device Driver Overview

- Block Devices
  - Random access devices (buffering)
  - File system은 random access이기 때문에 a block device상으로 mount될 수 있다.
  - Disk는 일반적으로 block device형태로 개발된다.
- Char devices
  - Sequential access (no buffering)
  - Printers, scanners, sound boards, etc.

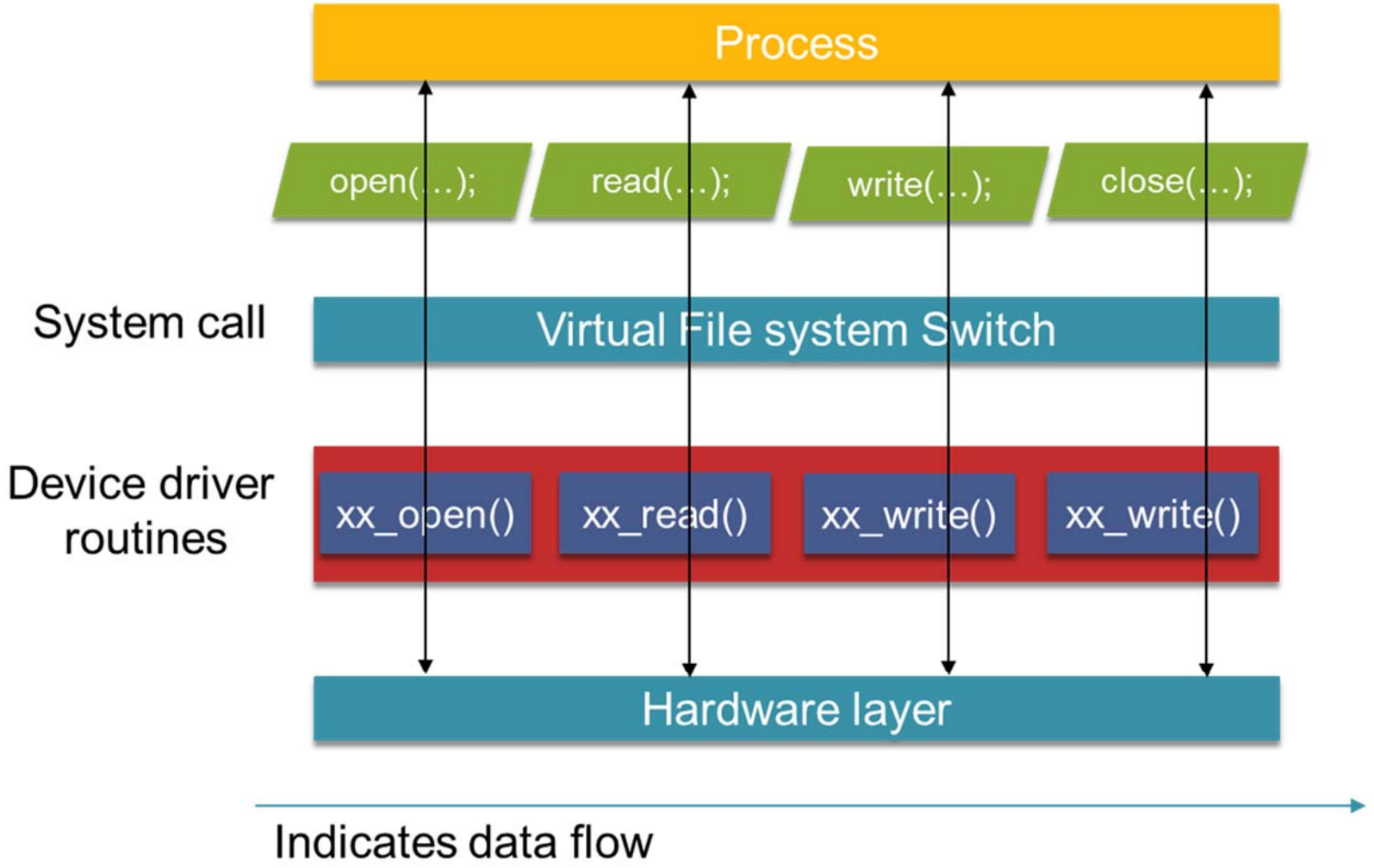
# Device Driver Flow

- Device Driver I/O Flow



# Device Driver Flow

- I/O Sub System



# Usage of Device Driver

- Special files
  - 모든 device는 리눅스 상에서 file의 형태로 존재
  - Device의 user-level interface를 special file이라 부른다.
  - 이러한 special file(often called device nodes)은 /dev directory안에 존재.
  - command ls -l /dev/lp* 를 수행하면 status information를 확인 할 수 있다.

```
crw-rw-rw 1 root root 6, 0 April 23 1994 /dev/lp0
```

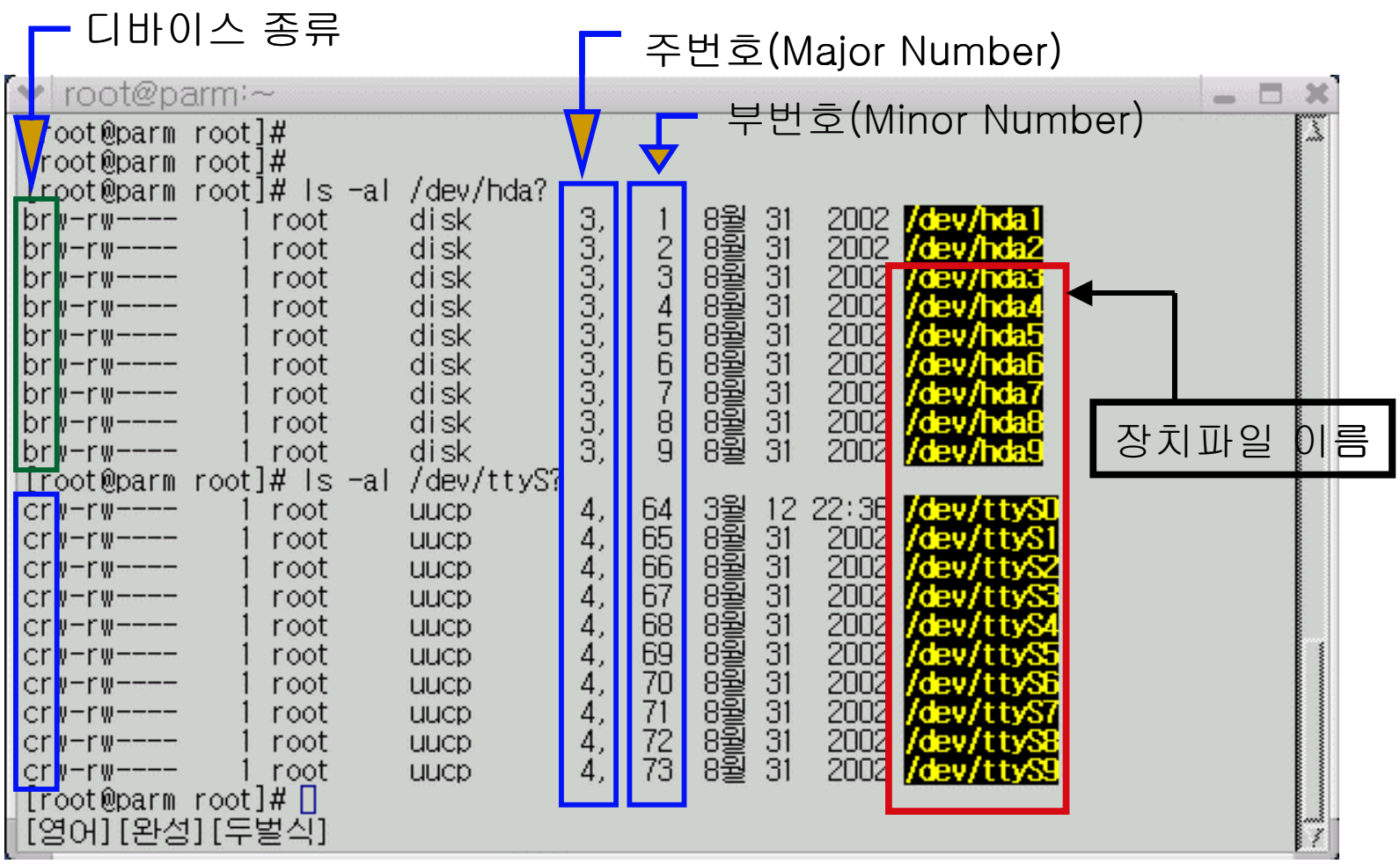
  - 위 예시는, lp0는 character type device (file mode field의 첫 시작 문자가 c)이며, the major number는 6, minor device number는 0 임을 의미한다.
  - mknod 명령으로 디바이스 드라이버에 접근할 수 있는 장치 파일 생성

# Usage of Device Driver

- Major number
  - 각 device driver는 device를 구분하기 위해서 unique major number를 부여.
  - 이는 Linux Device Registrar에 의해 할당
  - Driver정보를 포함하고 있는 배열의 index
  - # cat /proc/devices

- Minor number
  - device instance마다 부여된 번호
  - System이 다수의 IDE hard disk를 보유하고 있을 때 각각은 major number 3을 부여 받지만 minor number는 모두 다름.

# List of /dev directory



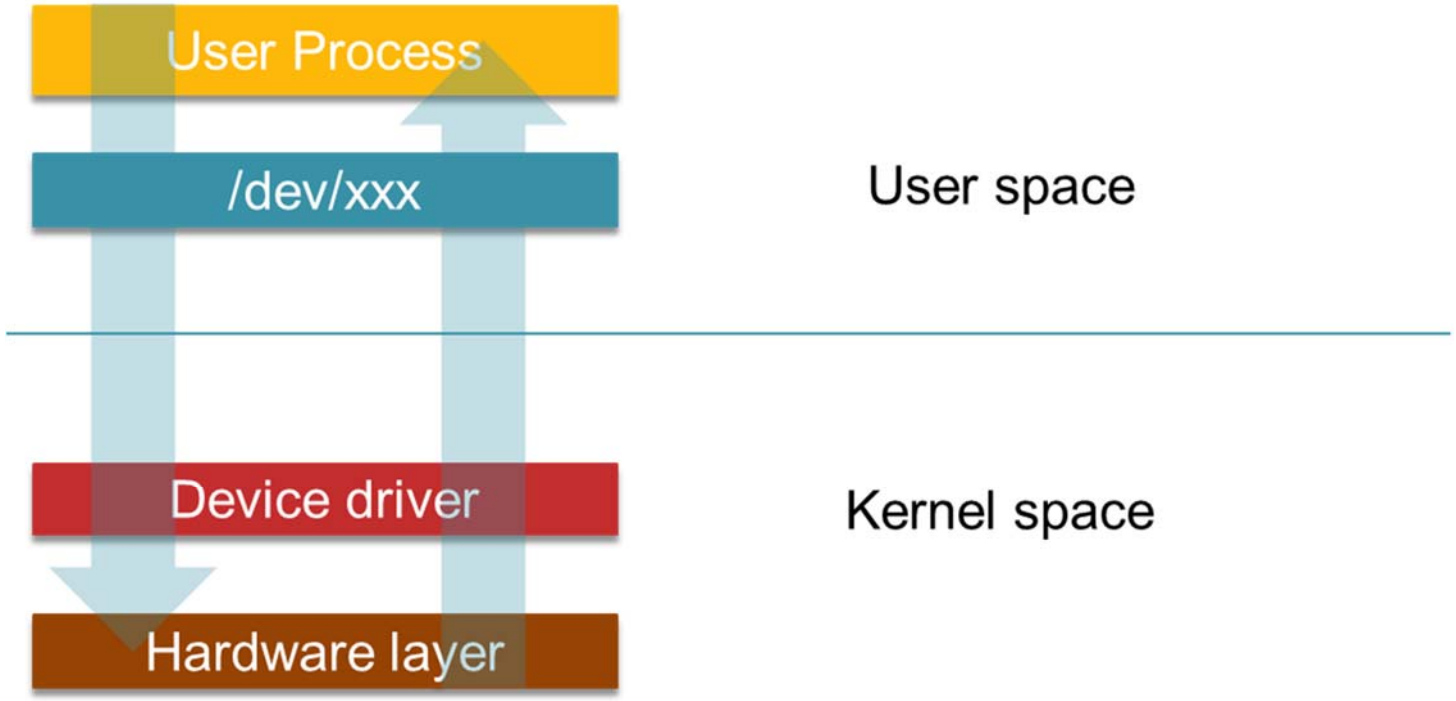
# Linux Major Number List

| Major | Character devices | Block devices |
|---|---|---|
| 0 | *unnamed for NFS, network and so on* | |
| 1 | Memory device(mem) | RAM disk |
| 2 | | Floppy disks(fd*) |
| 3 | | IDE hard disks(hd*) |
| 4 | Terminals | |
| 5 | Terminals & AUX | |
| 6 | Parallel Interfaces | |
| 7 | Virtual Consoles(vcs*) | |
| 8 | | SCSI hard disks(sd*) |
| 9 | SCSI tapes(st*) | |
| 10 | Bus mice(bm, psaux) | |
| 11 | | SCSI CD-ROM(scd*) |
| 12 | QIC02 tape | |
| 13 | PC Speaker driver | XT 8-bit hard disks(xd*) |
| 14 | Sound cards | BIOS hard disk support |
| 15 | Joystick | Cdu31a/33a CD-ROM |
| – | | |
| 19 | Cyclades drivers | Double compressing driver |
| 20 | Cyclades drivers | |
| 21 | SCSI generic | |
| 22 | | 2nd IDE interface driver |
| 23 | | Mitsumi CD-ROM(mcd*) |
| 24 | | Sony 535 CD-ROM |
| 25 | | Matsushita CD-ROM 1 |

참조
$LinuxSrc/Documentation/devices.txt
$LinuxSrc/include/linux/major.h

# Usage of Device Driver

- Device driver and special files



# Usage of Device Driver

- File operation structure
  - Device를 위한 file operation structure는 device를 위한 function pointer로 구성
  - 이는 generic interface상의 function을 specific device behavior에 연결.

- 이런 연결은 두 가지의 배열 상에서 유지:
  - static struct device_struct chrdevs[MAX_CHRDEV]
  - static struct device_struct blkdevs[MAX_BLKDEV]

- 같은 device는 character와 block interfaces를 모두 가질 수 있으며 각각의 배열에 저장된다.

# Usage of Device Driver

```
struct file_operations {
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char *, size_t, loff_t *);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
        int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, struct dentry *);
        int (*fasync) (int, struct file *, int);
        int (*check_media_change) (kdev_t dev);
        int (*revalidate) (kdev_t dev);
        int (*lock) (struct file *, int, struct file_lock *);
};
```

# include/linux/fs/h

fs.h (/work/achroimx6q/achroimx_kernel/include/linux) - gedit

Open Save Undo

fs.h

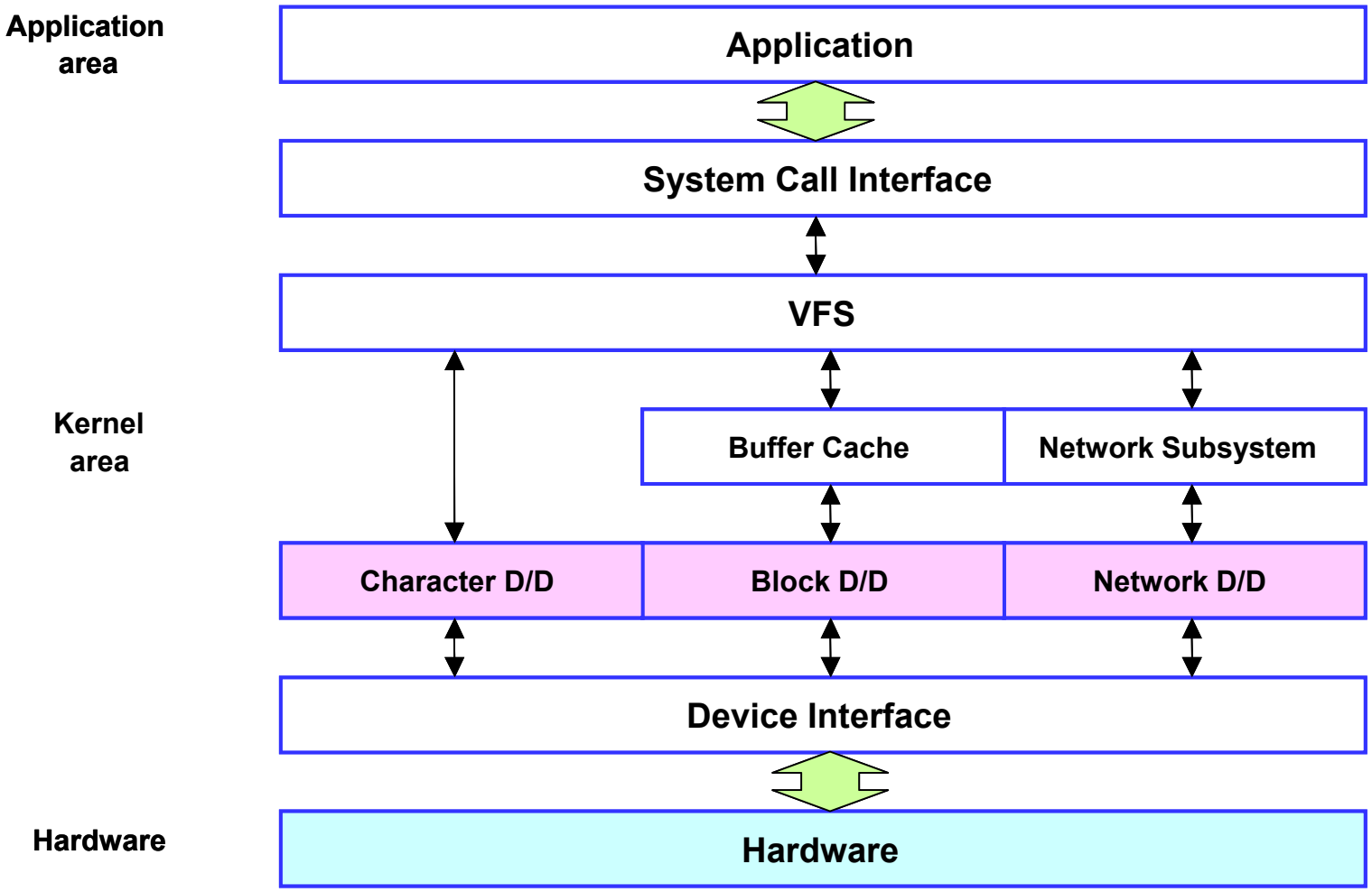
```
 * NOTE:
 * all file operations except setlease can be called without
 * the big kernel lock held in all filesystems.
 */
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*aio_read) (struct kiocb *, const struct iovec *, unsigned long, loff_t);
        ssize_t (*aio_write) (struct kiocb *, const struct iovec *, unsigned long, loff_t);
        int (*readdir) (struct file *, void *, filldir_t);
        unsigned int (*poll) (struct file *, struct poll_table_struct *);
        long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
        long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *, fl_owner_t id);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, int datasync);
        int (*aio_fsync) (struct kiocb *, int datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
        ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int);
        unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned long, unsigned
long);
        int (*check_flags)(int);
        int (*flock) (struct file *, int, struct file_lock *);
        ssize_t (*splice_write)(struct pipe_inode_info *, struct file *, loff_t *, size_t, unsigned int);
        ssize_t (*splice_read)(struct file *, loff_t *, struct pipe_inode_info *, size_t, unsigned int);
        int (*setlease)(struct file *, long, struct file_lock **);
        long (*fallocate)(struct file *file, int mode, loff_t offset,
                          loff_t len);
};
```

# Usage of Device Driver

- Char device driver
  - device driver 등록
    - extern int register_chrdev(unsigned int, const char *, struct file_operations *)
      - 첫 번째 parameter는 major number (0일 경우 사용하지 않는 번호를 동적으로 할당)
      - 두 번째 parameter는 device name
      - 세 번째 parameter는 file operation

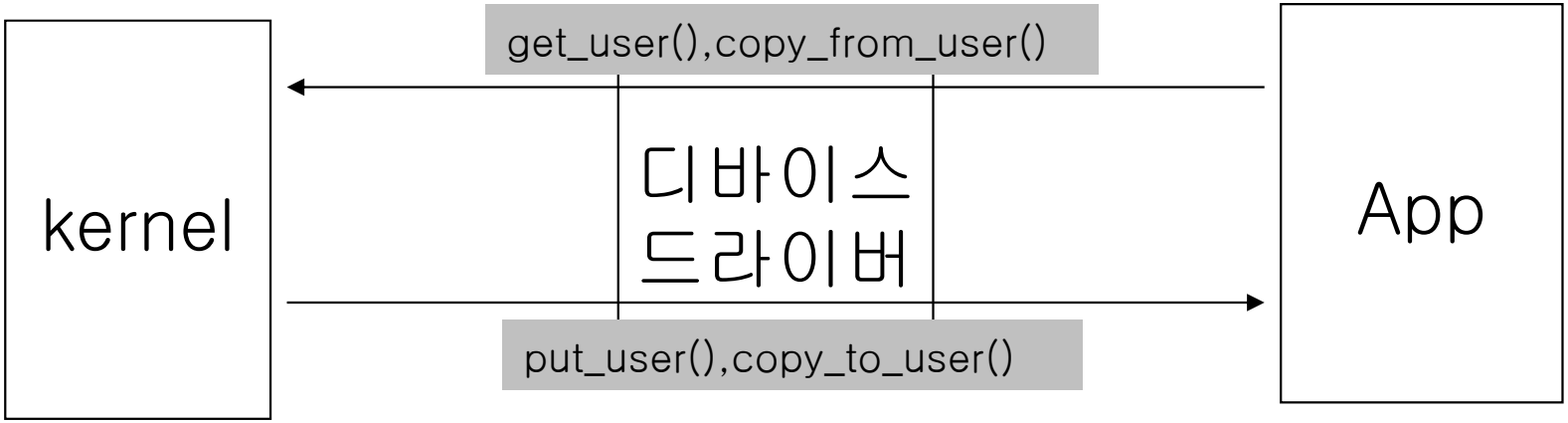
- Device driver 해제
  - extern int unregister_chrdev(unsigned int, const char *)
    - 첫 번째 parameter는 major number
    - 두 번째 parameter는 device name

# Linux Architecture



# DATA 전달 방법

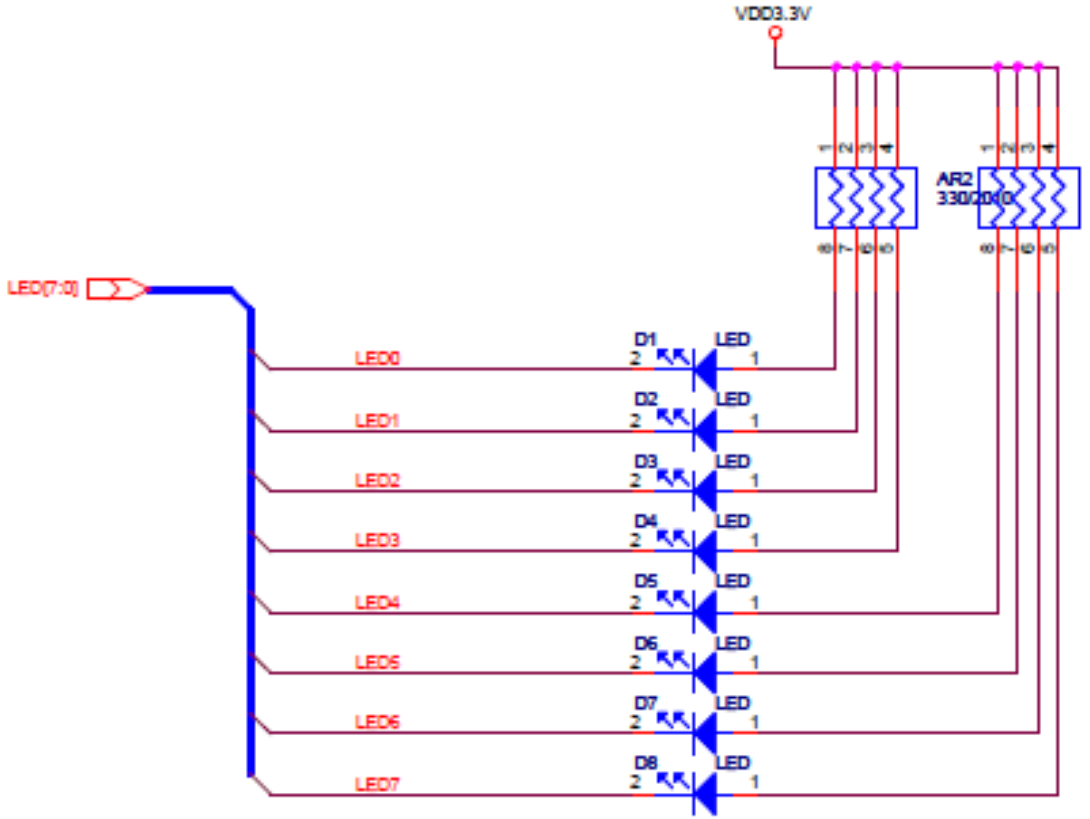
- 함수
  - get_user(void *, Const void *addr)
    - ✍사용자 공간의 어드레스 addr로 부터 변수x로 바이트를 복사
  - put_user(void *, Const void *addr)
    - ✍addr로부터 변수 x까지 사용자 공간으로 바이트를 복사
  - copy_to_user(void *to, void *from, unsinged long size)
  - copy_from_user(void *to, void *from, unsinged long size)
  - <asm/uaccess.h> 참조



# Device Driver Example

# FPGA LED Device

- FPGA LED Device Schematic



# FPGA LED Device

- FPGA LED Device Driver Code
  - Device definition

```
#define IOM_LED_MAJOR 260                 // ioboard led device major number
#define IOM_LED_NAME "fpga_led"           // ioboard led device name
#define IOM_LED_ADDRESS 0x08000016   // pysical address
```

  - Device File Operation

```
// define file_operations structure
struct file_operations iom_led_fops =
{
.owner = THIS_MODULE,
.open = iom_led_open,
.write = iom_led_write,
.read = iom_led_read,
.release = iom_led_release,
};
```

# FPGA LED Device

- FPGA LED Device Driver Code
  - Register Device Driver

```
int __init iom_led_init(void)
{
        int result;
        result = register_chrdev(IOM_LED_MAJOR, IOM_LED_NAME, &iom_led_fops);
        if(result < 0) {
                printk(KERN_WARNING"Can't get any major\n");
                return result;
        }
        iom_fpga_led_addr = ioremap(IOM_LED_ADDRESS, 0x1);
        printk("init module, %s major number : %d\n", IOM_LED_NAME, IOM_LED_MAJOR);
        return 0;
}
```

# FPGA LED Device

- FPGA LED Application Code
  - Open device and write data

```
dev = open(LED_DEVICE, O_RDWR);
if (dev<0) {
printf("Device open error : %s\n",LED_DEVICE);
exit(1);
}
retval=write(dev,&data,1);
```

# FPGA LED Device

- FPGA LED Makefile

```
#Makefile for a basic kernel module
obj-m := fpga_led_driver.o
KDIR :=/work/achroimx6q/kernel
PWD :=$(shell pwd)
all: driver app
#all: driver
driver:
	$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
app:
	arm-none-linux-gnueabi-gcc -o fpga_test_led fpga_test_led.c
install:
	cp -a fpga_led_driver.ko /nfsroot
	cp -a fpga_test_led /nfsroot
clean:
	rm -rf *.ko
	rm -rf *.mod.*
	rm -rf *.o
	rm -rf fpga_test_led
	rm -rf Module.symvers
	rm -rf modules.order
	rm -rf .led*
```

# FPGA LED Device

- make

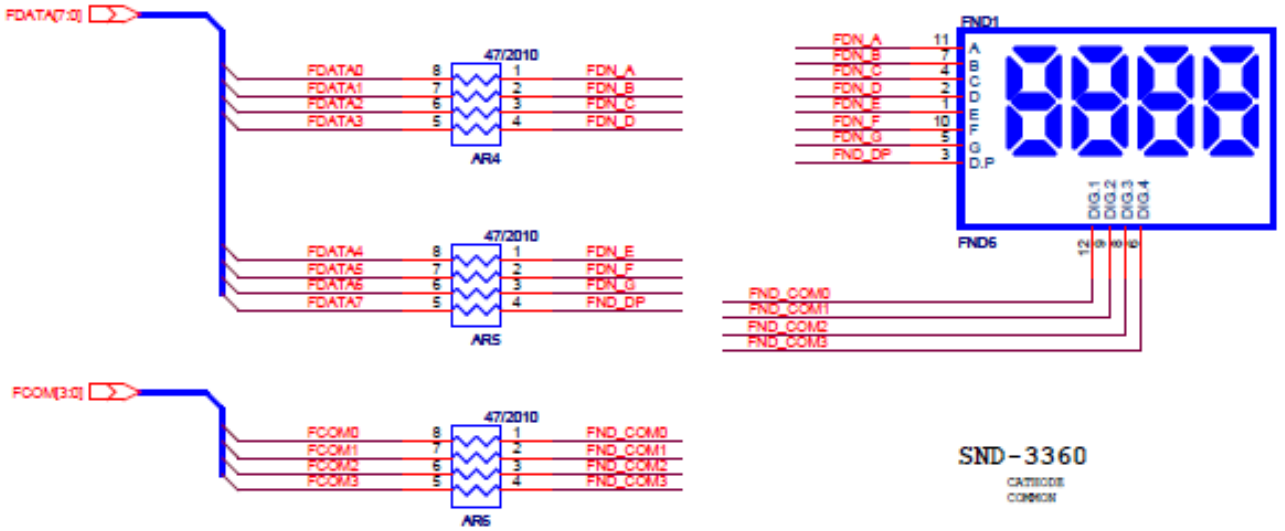
```
$ make
$ make install
```

- run on the target

```
# cd /mnt/nfs
# insmod fpga_led_driver.ko
# mknod /dev/fpga_led c 260 0
# ./fpga_test_led 1
```

# FPGA FND Device

- FPGA FND Device Schematic




| Digit | | | Address | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 / 2 | | | 0x0800_0004 | | | | | |
| 3 / 4 | | | 0x0700_0006 | | | | | |
| | | | | | | | | |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FND | D.P | G | F | E | D | C | B | A |

# FPGA FND Device

- FPGA FND Device Driver Code
  - Device definition

```
#define IOM_FND_MAJOR 261                 // ioboard fpga device major number
#define IOM_FND_NAME "fpga_fnd"           // ioboard fpga device name
#define IOM_FND_ADDRESS 0x08000004        // pysical address
```

  - Device File Operation

```
// define file_operations structure
struct file_operations iom_fnd_fops =
{
.owner    =        THIS_MODULE,
.open     =        iom_fpga_fnd_open,
.write    =        iom_fpga_fnd_write,
.read     =        iom_fpga_fnd_read,
.release  =        iom_fpga_fnd_release,
};
```

# FPGA FND Device

- FPGA FND Device Driver Code
  - Register Device Driver

```
int __init iom_fpga_fnd_init(void)
{
        int result;
        result = register_chrdev(IOM_FND_MAJOR, IOM_FND_NAME, &iom_fpg a_fnd_fops);
        if(result < 0) {
                printk(KERN_WARNING"Can't get any major\n");
                return result;
        }
        iom_fpga_fnd_addr = ioremap(IOM_FND_ADDRESS, 0x4);
        printk("init module, %s major number : %d\n", IOM_FND_NAME, IOM_FND_MAJOR);
        return 0;
}
```

# FPGA FND Device

- FPGA FND Device Driver Code
  - Write Data on Device Driver

```
// when write to fnd device ,call this function
ssize_t iom_fpga_fnd_write(struct file *inode, const char *gdata, size_t length, loff_t *off_what)
{
        int i;
        unsigned char value[4];
        unsigned short int value_short = 0;
        const char *tmp = gdata;
        if (copy_from_user(&value, tmp, 4))
                return -EFAULT;
value_short = value[0] << 12 | value[1] << 8 |value[2] << 4 |value[3];
outw(value_short,(unsigned int)iom_fpga_fnd_addr);
        return length;
}
```

# FPGA FND Device

- FPGA FND Application Code
  - Open device and write data

```
dev = open(FND_DEVICE, O_RDWR);
if (dev<0) {
printf("Device open error : %s\n",FND_DEVICE);
exit(1);
}
retval=write(dev,&data,4);
if(retval<0) {
printf("Write Error!\n");
return -1;
}
```

# FPGA FND Device

- FPGA FND Makefile

```
#Makefile for a basic kernel module
obj-m := fpga_fnd_driver.o
KDIR :=/work/achroimx6q/kernel
PWD :=$(shell pwd)
all: driver app
#all: driver
driver:
	$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
app:
	arm-none-linux-gnueabi-gcc -o fpga_test_fnd fpga_test_fnd.c
install:
	cp -a fpga_fnd_driver.ko /nfsroot
	cp -a fpga_test_fnd /nfsroot
clean:
	rm -rf *.ko
	rm -rf *.mod.*
	rm -rf *.o
	rm -rf fpga_test_fnd
	rm -rf Module.symvers
	rm -rf modules.order
	rm -rf .fnd*
	rm -rf .tmp*
```

# FPGA FND Device

- make

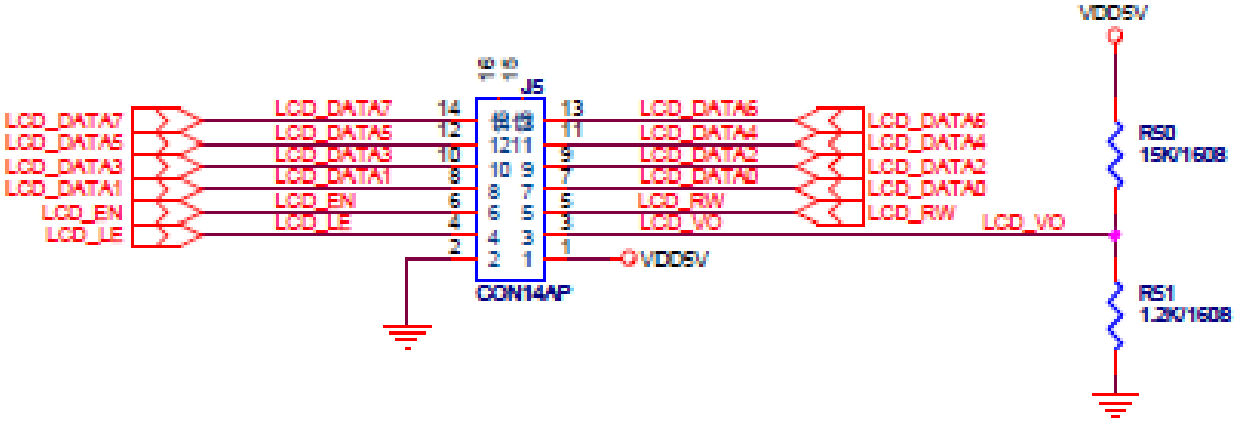
```
$ make
$ make install
```

- run on the target

```
# cd /mnt/nfs
# insmod fpga_fnd_driver.ko
# mknod /dev/fpga_fnd c 261 0
# ./fpga_test_led 1234
```

# FPGA Text LCD Device

- FPGA Text LCD Device Schematic



# FPGA Text LCD Device

- FPGA Text LCD Device Driver Code
  - Device definition

```
#define IOM_TEXT_LCD_MAJOR 263                  // ioboard fpga device major number
#define IOM_TEXT_LCD_NAME "fpga_text_lcd“       // ioboard fpga device name
#define IOM_TEXT_LCD_ADDRESS 0x08000090 // pysical address
```

  - Device File Operation

```
// define file_operations structure
struct file_operations iom_text_lcd_fops =
{
.owner    =        THIS_MODULE,
.open     =        iom_fpga_text_lcd_open,
.write    =        iom_fpga_text_lcd_write,
.release  =        iom_fpga_text_lcd_release,
};
```

# FPGA Text LCD Device

- FPGA Text LCD Device Driver Code
  - Register Device Driver

```
int __init iom_fpga_text_lcd_init(void)
{
        int result;
        result = register_chrdev(IOM_FPGA_TEXT_LCD_MAJOR, IOM_FPGA_TEXT_LCD_N
AME, &iom_fpga_text_lcd_fops);
        if(result < 0) {
                printk(KERN_WARNING"Can't get any major\n");
                return result;
        }
        iom_fpga_text_lcd_addr = ioremap(IOM_FPGA_TEXT_LCD_ADDRESS, 0x32);
        printk("init module, %s major number : %d\n", IOM_FPGA_TEXT_LCD_NAME, IOM_F
PGA_TEXT_LCD_MAJOR);
        return 0;
}
```

# FPGA Text LCD Device

- FPGA Text LCD Device Driver Code
  - Write Data on Device Driver

```
// when write to fpga_text_lcd device ,call this function
ssize_t iom_fpga_text_lcd_write(struct file *inode, const char *gdata, size_t length, loff_t *off_what)
{
        int i;
unsigned short int _s_value = 0;
        unsigned char value[32];
        const char *tmp = gdata;
        if (copy_from_user(&value, tmp, length))
                return -EFAULT;
        value[length]=0;
        for(i=0;i<length;i++)
{
_s_value = (value[i] & 0xFF) << 8 | value[i + 1] & 0xFF;
                outw(_s_value,(unsigned int)iom_fpga_text_lcd_addr+i);
i++;
}
        return length;
}
```

# FPGA Text LCD Device

- FPGA Text LCD Application Code
  - Open device and write data

```
str_size=strlen(argv[1]);
if(str_size>0) {
strncat(string,argv[1],str_size);
memset(string+str_size,' ',LINE_BUFF-str_size);
}
str_size=strlen(argv[2]);
if(str_size>0) {
strncat(string,argv[2],str_size);
memset(string+LINE_BUFF+str_size,' ',LINE_BUFF-str_size);
}
write(dev,string,MAX_BUFF);
```

# FPGA Text LCD Device

- FPGA Text LCD Makefile

```
#Makefile for a basic kernel module
obj-m := fpga_text_lcd_driver.o
KDIR :=/work/achroimx6q/kernel
PWD :=$(shell pwd)
APP := fpga_test_text_lcd
all: driver app
#all: driver
driver:
$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
app:
arm-none-linux-gnueabi-gcc -o $(APP) $(APP).c
install:
cp -a fpga_text_lcd_driver.ko /nfsroot
cp -a $(APP) /nfsroot
clean:
rm -rf *.ko
rm -rf *.mod.*
rm -rf *.o
rm -rf $(APP)
rm -rf Module.symvers
rm -rf modules.order
rm -rf .tmp*
rm -rf .fpga*
```

# FPGA Text LCD Device

- make

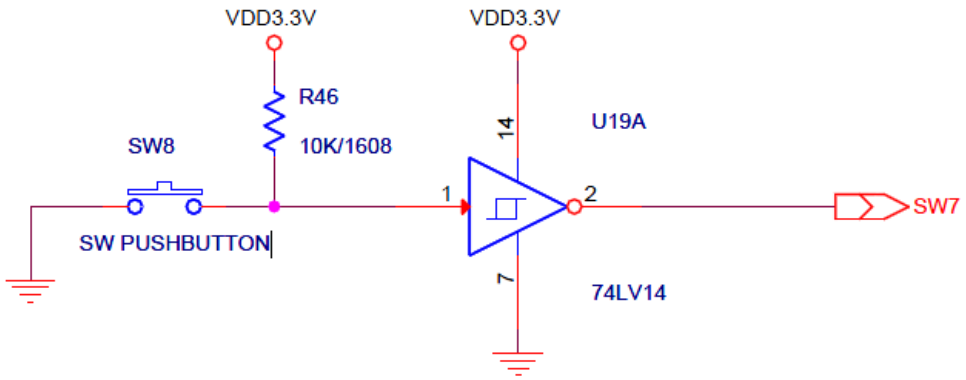
```
$ make
$ make install
```

- run on the target

```
# cd /mnt/nfs
# insmod fpga_text_lcd_driver.ko
# mknod /dev/fpga_text_lcd c 263 0
# ./fpga_test_text_lcd hello world
```

# FPGA Push Switch Device

- FPGA Push Switch Device Schematic



# FPGA Push Switch Device

- FPGA Push Switch Device Driver Code
  - Device definition

```
#define IOM_FPGA_PUSH_SWITCH_MAJOR 265 // ioboard led device major number
#define IOM_FPGA_PUSH_SWITCH_NAME "fpga_push_switch“ // ioboard led device name
#define IOM_FPGA_PUSH_SWITCH_ADDRESS 0x08000050        // pysical address
```

  - Device File Operation

```
// define file_operations structure
struct file_operations iom_fpga_push_switch_fops =
{
        owner:      THIS_MODULE,
        open:       iom_fpga_push_switch_open,
        read:       iom_fpga_push_switch_read,
        release:    iom_fpga_push_switch_release,
};
```

# FPGA Push Switch Device

- FPGA Push Switch Device Driver Code
  - Register Device Driver

```
int __init iom_fpga_push_switch_init(void)
{
        int result;
        result = register_chrdev(IOM_FPGA_PUSH_SWITCH_MAJOR, IOM_FPGA_PUSH_S
WITCH _NAME, &iom_fpga_push_switch_fops);
        if(result < 0) {
                printk(KERN_WARNING"Can't get any major\n");
                return result;
        }
        iom_fpga_push_switch_addr = ioremap(IOM_FPGA_PUSH_SWITCH_ADDRESS, 0x1
8);
        printk("init module : %s major number : %d\n", IOM_FPGA_PUSH_SWITCH_NAME, I
OM_FPGA_PUSH_SWITCH_MAJOR);
        return 0;
}
```

# FPGA Push Switch Device

- FPGA Push Switch Device Driver Code
  - Read Data on Device Driver

```
// when read from fpga_push_switch device ,call this function
ssize_t iom_fpga_push_switch_read(struct file *inode, char *gdata, size_t length, loff_t *off_what)
{
        int i;
        unsigned char push_sw_value[MAX_BUTTON];
        unsigned short int _s_value;
        for(i=0;i<length;i++)
{
                _s_value = inw((unsigned int)iom_fpga_push_switch_addr+i*2);
                push_sw_value[i] = _s_value &0xFF;
}
        if (copy_to_user(gdata, &push_sw_value, length))
                return -EFAULT;
        return length;
}
```

# FPGA Push Switch Device

- FPGA Push Switch Application Code
  - Open device and read data

```
dev = open("/dev/fpga_push_switch", O_RDWR);
if (dev<0){
    printf("Device Open Error\n");
    close(dev);
    return -1;
}
(void)signal(SIGINT, user_signal1);
buff_size=sizeof(push_sw_buff);
printf("Press <ctrl+c> to quit. \n");
while(!quit){
    usleep(400000);
    read(dev, &push_sw_buff, buff_size);
    for(i=0;i<MAX_BUTTON;i++) {
    printf("[%d] ",push_sw_buff[i]);
}
printf("\n");
}
```

# FPGA Push Switch Device

- FPGA Push Switch Makefile

```
#Makefile for a basic kernel module
obj-m := fpga_push_switch_driver.o
KDIR :=/work/achroimx6q/kernel
PWD :=$(shell pwd)
all: driver app
#all: driver
driver:
	$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
app:
	arm-none-linux-gnueabi-gcc -o fpga_test_push_switch fpga_test_push_switch.c
install:
	cp -a fpga_push_switch_driver.ko /nfsroot
	cp -a fpga_test_push_switch /nfsroot
clean:
	rm -rf *.ko
	rm -rf *.mod.*
	rm -rf *.o
	rm -rf fpga_test_push_switch
	rm -rf Module.symvers
	rm -rf .tmp*
	rm -rf .fpga*
	rm -rf modules.order
```

# FPGA Push Switch Device

- make

```
$ make
$ make install
```

- run on the target

```
# cd /mnt/nfs
# insmod fpga_push_switch_driver.ko
# mknod /dev/fpga_push_swicth c 265 0
# ./fpga_test_push_switch
```

# FPGA Push Switch Device

X ∨ ∧ root@ubuntu

File Edit View Terminal Help

```
root@AchroIMX6Q nfs$ insmod fpga_push_switch_driver.ko
init module, fpga_push_switch major number: 265
root@AchroIMX6Q nfs$ mknod /dev/fpga_push_switch c 265 0
root@AchroIMX6Q nfs$ ./fpga_test_push_switch
Press <ctrl+c> to quit
[0] [0] [0] [0] [0] [0] [0] [0] [0]
[0] [0] [0] [0] [0] [0] [0] [0] [0]
[0] [0] [0] [0] [0] [0] [0] [0] [0]
```