망고100 보드로 놀아보자-14

디바이스 드라이버 작성 기초

http://cafe.naver.com/embeddedcrazyboys

http://www.mangoboard.com

Device Driver 작성(1)

- Device structure
  - 디바이스 구조체 : 2 개의 필드로 구성된 구조체
    - Name field
    - file_operation files

```
static struct char_device_struct {
        struct char_device_struct *next;
        unsigned int major;
        unsigned int baseminor;
        int minorct;
        char name[64];
        struct cdev *cdev;              /* will die */
} *chrdevs[CHRDEV_MAJOR_HASH_SIZE];

struct file_operations { /* include/linux/fs.h */
    lseek;
    read;
    write;
    readdir;
    poll;
    ioctl;
    mmap;
    open;
    flush;
    release;
...
}
```

Device Driver 작성(2)

- 디바이스 드라이버 등록
  - 드라이버를 커널에 등록하고, 파일 연산을 정의하는 등의 초기화 작업 수행이 필요
  - 모듈의 형태에서는 init_module()함수에서 초기화 수행
  - 드라이버의 등록 함수

```
int register_chrdev( unsigned int major,
                     const * name,
                     struct file_operations * fops);
```

    - 커널에 지정되어 있는 chrdevs 구조에 새로운 char device 등록
      - major number : 주번호, 0을 주면 사용하지 않는 값을 반환
      - name : 디바이스 이름으로 /proc/devices에 나타남
      - fops: 디바이스와 연관된 파일 연산 구조체 포인터
    - 음수가 반환되면 오류가 발생했음을 나타냄

Device Driver 작성(3)

- 디바이스 드라이버 제거
  - 더 이상 사용되지 않는 드라이버의 제거
  - Rmmod하는 명령을 이용하여 제거하며, 이때 드라이버 내의 cleanup_module이 호출되는데, 이 루틴 안에서 다음의 시스템 콜을 호출
  - 드라이버 제거 함수

```
int unregister_chrdev( unsigned int major, const * name);
```

Device Driver 작성(4)

- 파일 연산
  - 디바이스 드라이버를 일반적인 파일과 유사한 인터페이스를 이용하여 관리
    - 각 디바이스는 파일 형태로 존재하고, 커널은 파일 연산을 이용하여 I/O 연산을 수행하도록 인터페이스 구성
    - 디바이스 드라이버를 구현한다는 것은 상당부분 파일연산 구조체에서 요구되는 기능들을 프로그래밍 한다는 것을 의미
    - 가상의 dummy character device 구현예제에서의 파일연산 구조체 정의 예

Device Driver 작성(5)

- 파일 연산 구조체의 전체 구조

```
struct file_operations {  /* <linux/fs.h> */
        struct module *owner;
        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 datasync);
        int (*fasync) (int, struct file *, int);
        int (*lock) (struct file *, int, struct file_lock *);
        ssize_t (*readv) (struct file *, const struct iovec *, unsigned long, loff_t *);
        ssize_t (*writev) (struct file *, const struct iovec *, unsigned long, loff_t *);
};
```

Device Driver 작성(6)

- File operations

loff_t (*__llseek__)(struct file *, loff_t, int);

→현재의 read/write 위치를 옮긴다.

ssize_t (*__read__)(struct file *, char *, size_t, loff_t *);

→디바이스에서 데이터를 가져오기 위해서 사용

ssize_t (*__write__)(struct file*, const char*, size_t, loff_t*);

→디바이스에 데이터를 쓰기 위해서 사용

int (*__readdir__)(struct file *, void *, filldir_t);

→디렉토리를 다룰 때 사용

unsigned int (*__poll__)(struct file*, struct poll_table_struct*);

→현재 프로세스를 대기 큐에 넣기 위해서 사용

Device Driver 작성(7)

- File operations

int (*__ioctl__) (struct inode *, struct file *, unsigned int, unsigned long);

→디바이스에 종속적인 명령을 만들기 위해 사용

Int (*__mmap__) (struct file *, struct vm_area_struct *);

→디바이스 메모리를 프로세스 메모리에 매핑

int (*__open__) (struct inode *, struct file *);

→디바이스 노드에 의해 수행되는 첫번째 동작

int (*__flush__) (struct file *);

→디바이스를 닫기 전에 모든 데이터를 쓴다.

int (*__release__) (struct inode *, struct file *);

→디바이스를 닫을 때 수행

Device Driver 작성(8)

- File operations

int (*__fsync__) (struct file *, struct dentry *);

→버퍼에 있는 데이터를 모두 디바이스에 쓴다

int (*__fasync__) (int, struct file *, int);

int (*__check_media_change__) (kdev_t dev);

→블록 디바이스에서 사용, 제거 가능한 미디어에 사용

int (*__revalidate__) (kdev_t dev);

→블록 디바이스에서 사용, 버퍼 캐쉬의 관리와 상관

int (*__lock__) (struct file *, int, struct file_lock *);

→파일에 lock을 걸기 위해서 사용

Device Driver 작성(9)

- Dummy Character device 드라이버 소스코드

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/errno.h>
#include <linux/types.h>
#include <linux/fcntl.h>

#define DUMMY_MAJOR_NUMBER 254  /* dummy.c */
#define DEV_NAME "dummy-device"
int dummy_open(struct inode *inode,struct file *filp)
{
        int num=MINOR(inode->i_rdev);
        printk(" Open call for Dummy Char Device  call open ->minor:%d\n",num);
        return 0;
}
loff_t dummy_llseek(struct file *filp,loff_t off,int whence)
{
        printk("call llseek->off:%08x, whenec:%08x\n",off,whence);
        return 0x23;
}
```

Device Driver 작성(10)

- Dummy Character device 드라이버 소스코드

```
ssize_t dummy_read(struct file *filp,char *buf, size_t count,loff_t *f_pos)
{
	printk("call read ->buf:%o8x, count:%o8x\n",buf,count);
	return ox33;
}
ssize_t dummy_write(struct file *filp,const char *buf, size_t count,loff_t *f_pos)
{
	printk("call write->buf:%o8x, count:%o8x\n",buf,count);
	return ox43;
}
int dummy_ioctl(struct inode *inode, struct file *filp,unsigned int cmd,unsigned long arg)
{
	printk("call ioctl->cmd:%o8,arg:%o8x\n",cmd,arg);
	return ox53;
}
int dummy_release(struct inode *inode, struct file *filp)
{
	printk(" Release call for Dummy Char Device \n");
	return o;
}
```

Device Driver 작성(11)

- Dummy Character device 드라이버 소스코드

```
struct file_operations dummy_fops =
{
	.owner =THIS_MODULE,
	.llseek=dummy_llseek,
	.open = dummy_open,
	.read=dummy_read,
	.write=dummy_write,
	.ioctl=dummy_ioctl,
	.release=dummy_release,

};
int dummy_init(void)
{
	int result;
	printk("call dummy_init\n");
	result=register_chrdev(DUMMY_MAJOR_NUMBER,DEV_NAME,&dummy_fops);
	if(result<0) return result;
	return 0;
}
void dummy_exit(void)
{
	printk("call dummy_exit\n");
	unregister_chrdev(DUMMY_MAJOR_NUMBER,DEV_NAME);
}
module_init(dummy_init);
module_exit(dummy_exit);

MODULE_LICENSE("GPL");
```

Device Driver 작성(12)

- System call : dummy_open
  - File_operations 구조체에서 open operation 구현
  - Application program에서 'open' 에 의해서 불려짐

```
int dummy_open(struct inode *inode, struct file *file)
{
        printk("Open call for Dummy Char Device \n");
        return 0;
}
```

- System call : dummy_release
  - File_operations 구조체에서 release operation 구현
  - Application program에서 'close' 에 의해서 불려짐

```
int dummy_release(struct inode *inode, struct file *file)
{
        printk("Release call for Dummy Char Device \n");
        return 0;
}
```

Device Driver 작성(11)

- System call : dummy_read
  - File_operations 구조체에서 read operation 구현
  - Application program에서 'read' 에 의해서 불려짐

```
ssize_t dummy_read(struct file *file, char *buffer,   size_t length, loff_t *offset)
{
      printk("Read Call for Dummy Device \n");
      buffer[0] = 0x34;     return 0;
}
```

- System call : dummy_write
  - File_operations 구조체에서 write operation 구현
  - Application program에서 'write' 에 의해서 불려짐

```
ssize_t dummy_write(struct file *file, const char *buffer, size_t length,  loff_t *offset)
{
      printk("Write Call  for Dummy Device :  [%x]\n ", buffer[0]);
      return 0;
}
```

드라이버 컴파일/로딩/노드 생성

- 디바이스 드라이버 컴파일
- Makefile 작성

```
obj-m:=dummy-driver.o
KDIR:=../mango100_kernel_2010_06_30
PWD:=$(shell pwd)

default:
	$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
clean:
	rm -rf *.ko
	rm -rf *.mod.*
	rm -rf .*.cmd
	rm -rf *.o
```

- #make

module 파일시스템에 포함방법 (저장장치를 이용하는 방법)

- Usb stick, MMC 등 저장장치를 HOST PC에 삽입
- #df
- #cp dummy-driver.ko /mount 디렉토리
- HOST PC 저장장치 분리, 망고 보드에 삽입
- #insmod dummy-driver.ko

```
root@Mango:~# insmod dummy-driver.ko
call dummy_init
insmod: error inserting 'dummy-driver.ko': -1 Device or resource busy
```

에러 발생 시 #ls /sys/dev/char |grep 주번호, 확인 후 소스에서 major 번호 수정

```
root@Mango:/sys/dev/char# ls -al | grep 254
lrwxrwxrwx    1 root     root             0 Apr 17 00:49 10:254 -> ../../class/mi
sc/s3c-jpg
lrwxrwxrwx    1 root     root             0 Apr 17 00:49 254:0 -> ../../class/rtc
/rtc0
lrwxrwxrwx    1 root     root             0 Apr 17 00:49 2:254 -> ../../class/tty
/ptyee
lrwxrwxrwx    1 root     root             0 Apr 17 00:49 3:254 -> ../../class/tty
/ttyee
```

- #cat /proc/modules

장치파일 등록(1/2)

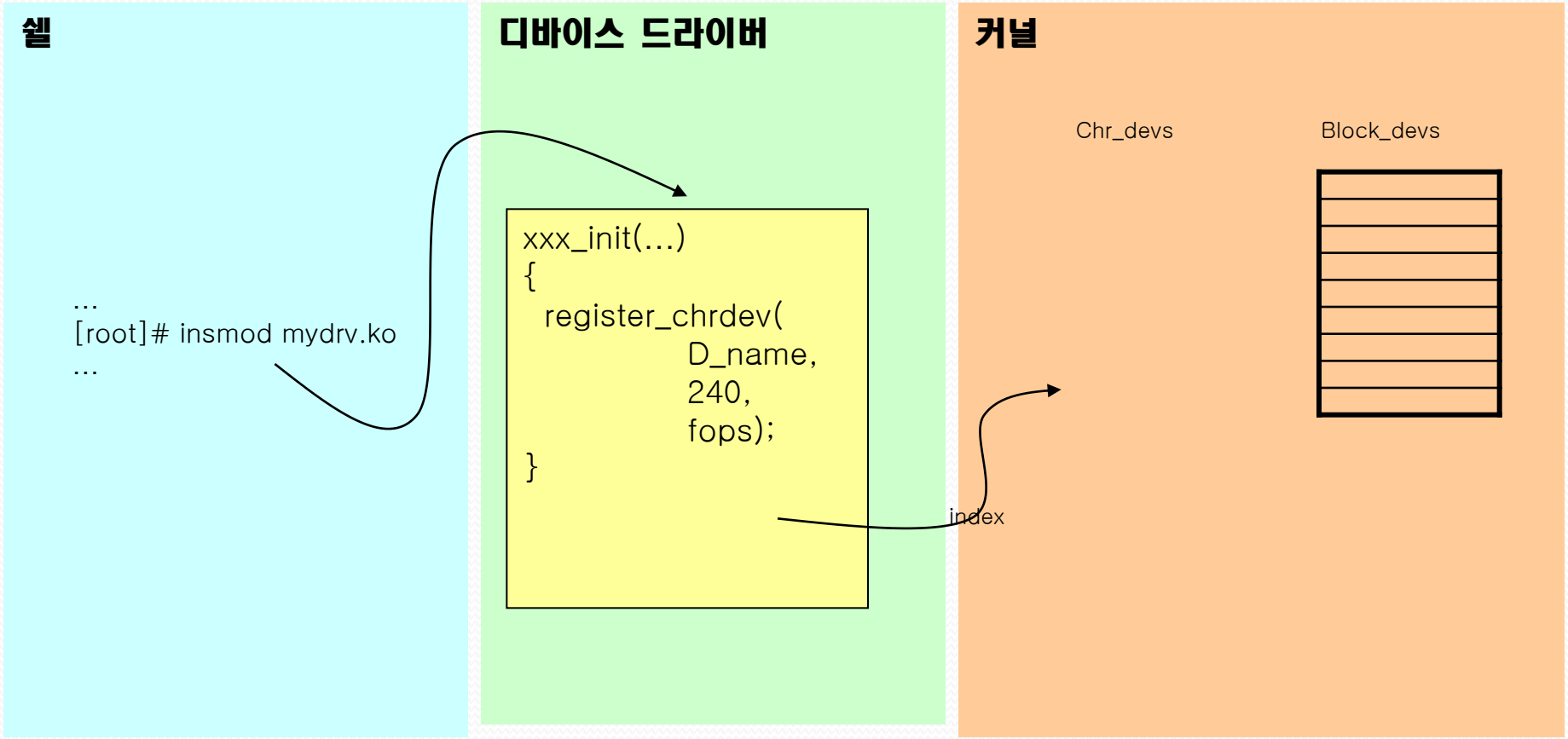
- 장치 파일 등록

```
[root]# insmod mydrv_dev.ko
```

```
Int xxx_init(void)
 {
     int result;
       result = register_chrdev(RDWR_DEV_MAJOR, RDWR_DEV_NAME, &xxx_fops);
       .....
       return 0;
 }
  module_init(xxx_init);
```

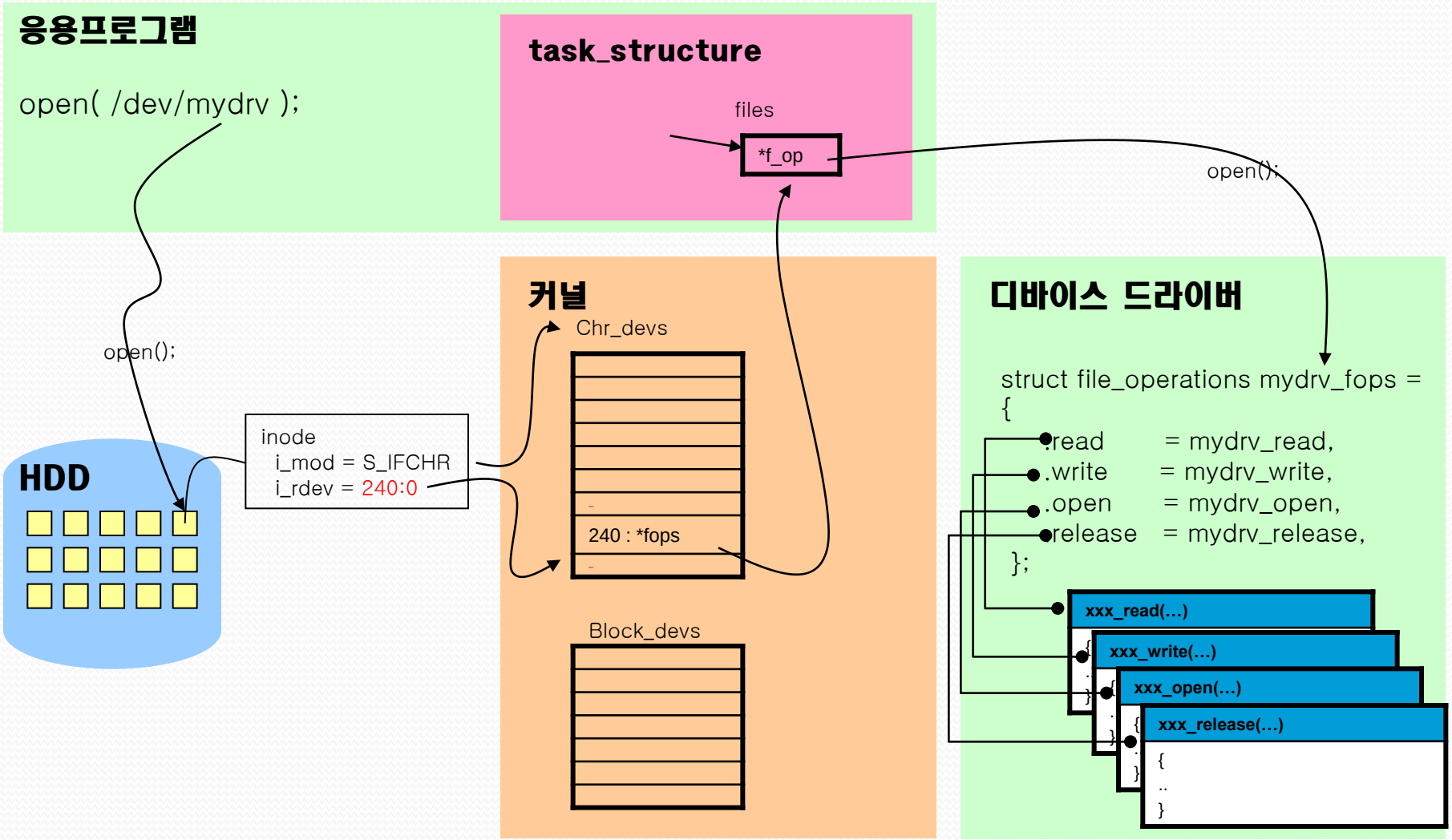
장치파일 등록(2/2)

- register_chrdev 의 기능



Open 의 이해

- Open 과정



Application Program 작성

- Read / write application program 작성
  - 작성한 dummy character device를 테스트
  - Dummy-device를 열고 문자열을 read /write 함
  - Console의 커널 내부 정보를 통해 실제 dummy_device 를 통해서 read / write가 이루어졌는지 확인

Application Program 작성

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <unistd.h>
#define DEVICE_FILENAME "/dev/dummy-driver”
int main(){
        int dev;
        char buff[128];
        int ret;

        printf("1) device file open\n");

        dev = open(DEVICE_FILENAME, O_RDWR|O_NDELAY);
        if(dev>=0)
        {
                printf("2) seek function call\n");

                ret = lseek(dev, 0x20, SEEK_SET);
                printf("ret = %08X\n",ret);

                printf("3) read function call\n");

                ret = read(dev,0x30, 0x31);
                printf("ret = %08X\n", ret);

                printf("4) write function call\n");
                ret = write(dev, 0x40, 0x41);
                printf("ret = %08X\n",ret);
                printf("5) ioctl function call\n");
                ret = ioctl(dev, 0x51, 0x52);
                printf("ret = %08X\n" ,ret);

                printf("6) device file close\n");
                ret = close(dev);
                printf("ret = %08X\n" ,ret);
        }
        return 0;
}
```

Application Program 컴파일/적재

- #arm-linux-gcc –o dummy-app.o dummy-app.c
- Dummy-app.o파일을 망고보드 파일 시스템에 복사
- #cp dummy-app.o /nfsroot
- #/sbin/mount –t nfs –o nolock 192.168.0.10:/nfsroot /mnt/nfs
- #cp /mnt/nfs/dummy-app.o ~

Application 실행 방법

- #mknod /dev/dummy-driver c 240 32
- #./dummy-app.o

```
root@Mango:/sys/module# mknod /dev/dummy-driver c 240 32
root@Mango:/sys/module# cd /mnt/nfs
root@Mango:/mnt/nfs# ./dummy-app.o
1) device file opencall open ->minor:32
call llseek->off:00000000, whenec:00000020
call read ->buf:00000030, count:00000031
call write->buf:00000040, count:00000041
call ioctl->cmd:%,arg:00000051
call release

2) seek function call
ret = 00000023
3) read function call
ret = 00000033
4) write function call
ret = 00000043
5) ioctl function call
ret = 00000053
6) device file close
ret = 00000000
```