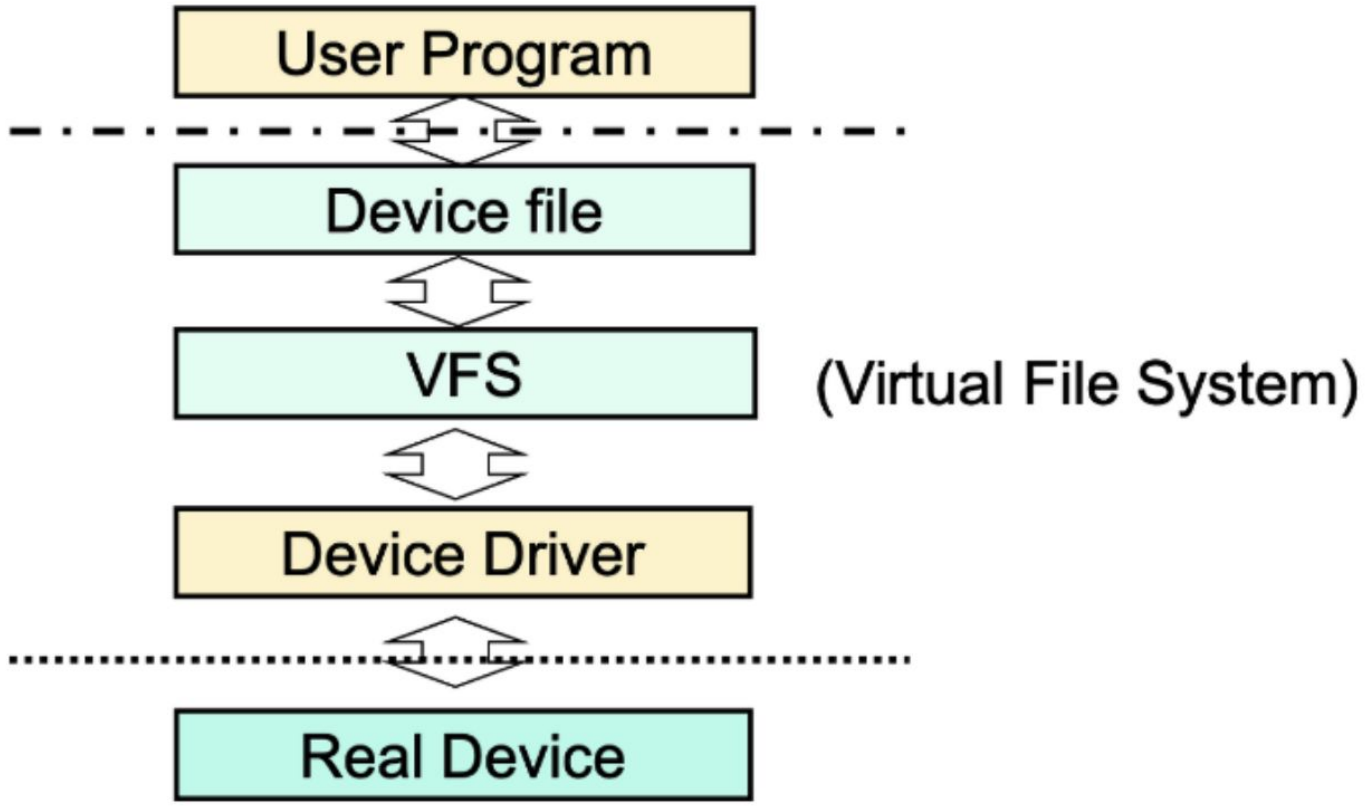
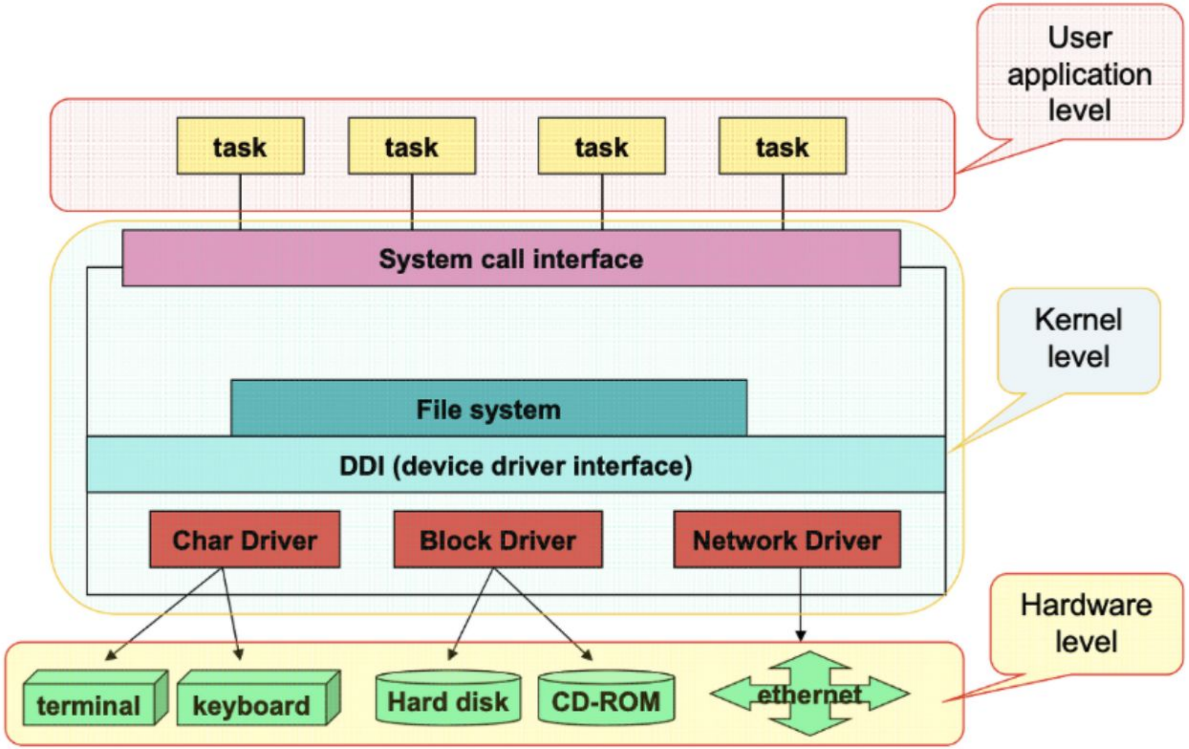
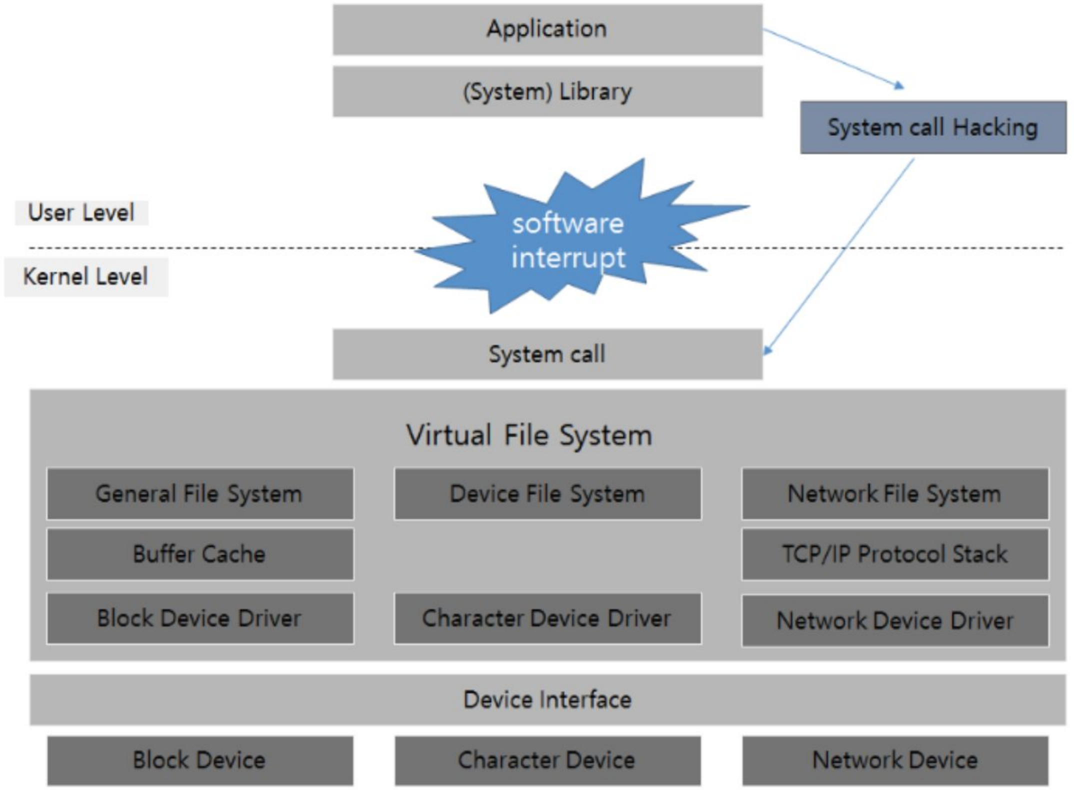
시스템프로그램 : Linux
디바이스 드라이버 기초

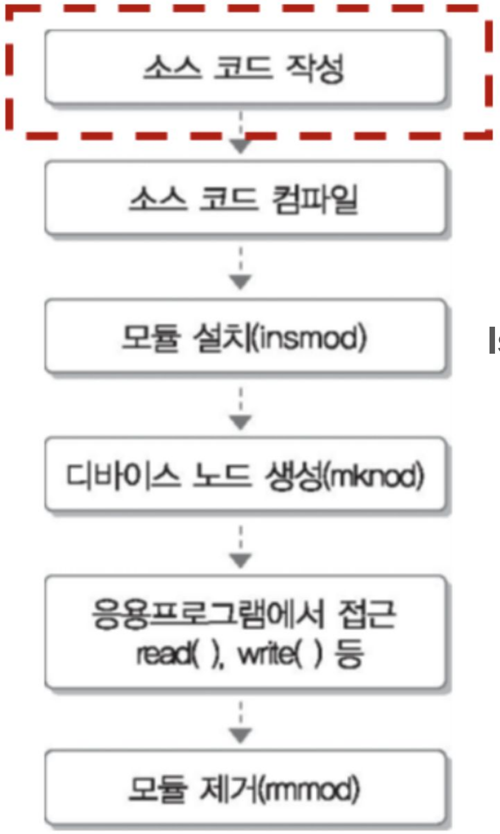

User Program
Device file
VFS
(Virtual File System)
Device Driver
Real Device


User application level
task
task
task
task
System call interface
Kernel level
File system
DDI (device driver interface)
Char Driver
Block Driver
Network Driver
Hardware level
terminal
keyboard
Hard disk
CD-ROM
ethernet

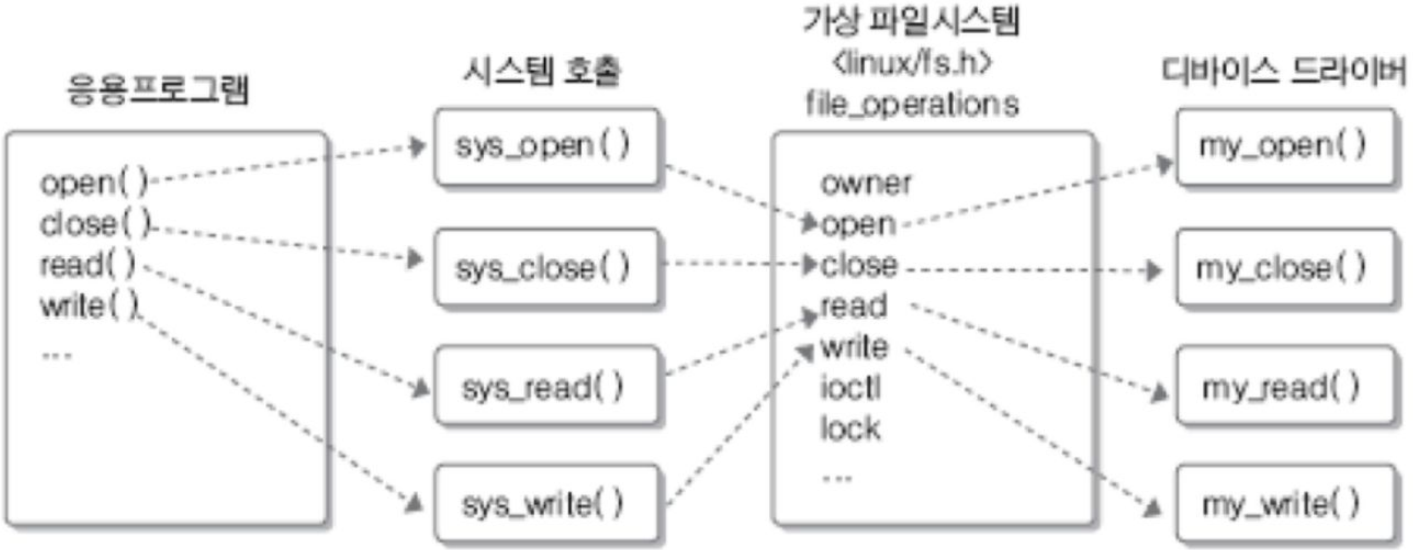
디바이스 드라이버

```
brw-rw----   1 root     disk      8,   1 Nov 15 10:52 sda1
crw-rw----+  1 root     cdrom    21,   0 Nov 15 10:52 sg0
crw-rw----   1 root     disk     21,   1 Nov 15 10:52 sg1
drwxrwxrwt   2 root     root          40 Nov 15 10:52 shm
crw-------   1 root     root     10, 231 Nov 15 10:52 snapshot
drwxr-xr-x   3 root     root         200 Nov 15 10:52 snd
brw-rw----+  1 root     cdrom    11,   0 Nov 15 10:52 sr0
lrwxrwxrwx   1 root     root          15 Nov 15 10:52 stderr -> /proc/self/fd
lrwxrwxrwx   1 root     root          15 Nov 15 10:52 stdin -> /proc/self/fd/
lrwxrwxrwx   1 root     root          15 Nov 15 10:52 stdout -> /proc/self/fd
crw-rw-rw-   1 root     tty       5,   0 Nov 15 20:05 tty
crw--w----   1 root     tty       4,   0 Nov 15 10:52 tty0
crw--w----   1 gdm      tty       4,   1 Nov 15 10:52 tty1
crw--w----   1 root     tty       4,  10 Nov 15 10:52 tty10
crw--w----   1 root     tty       4,  11 Nov 15 10:52 tty11
crw--w----   1 root     tty       4,  12 Nov 15 10:52 tty12
crw--w----   1 root     tty       4,  13 Nov 15 10:52 tty13
crw--w----   1 root     tty       4,  14 Nov 15 10:52 tty14
```


Application
(System) Library
System call Hacking
User Level
software interrupt
Kernel Level
System call
Virtual File System
General File System
Device File System
Network File System
Buffer Cache
TCP/IP Protocol Stack
Block Device Driver
Character Device Driver
Network Device Driver
Device Interface
Block Device
Character Device
Network Device


소스 코드 작성
소스 코드 컴파일
모듈 설치(insmod)
디바이스 노드 생성(mknod)
응용프로그램에서 접근
read( ), write( ) 등
모듈 제거(rmmod)

lsmod : 목록 보기


응용프로그램
open( )
close( )
read( )
write( )
...
시스템 호출
sys_open( )
sys_close( )
sys_read( )
sys_write( )
가상 파일시스템
<linux/fs.h>
file_operations
owner
open
close
read
write
ioctl
lock
...
디바이스 드라이버
my_open( )
my_close( )
my_read( )
my_write( )

```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <asm/uaccess.h>
#include <linux/slab.h>

static char *buffer = NULL;

int test_open(struct inode *inode, struct file *filp) {
    printk(KERN_ALERT "test_device open function called\n");
    return 0;
}

int test_device_release(struct inode *inode, struct file *filp) {
    printk(KERN_ALERT "testdevice release function called\n");
    return 0;
}
```

```
ssize_t test_device_write(struct file *filp, const char *buf, size_t count, loff_t *f_pos) {
    printk(KERN_ALERT "test_device write function called\n");
    strcpy(buffer, buf);
    return count;
}

ssize_t test_device_read(struct file *filp, char *buf, size_t count, loff_t *f_pos) {
    printk(KERN_ALERT "test_device read function called\n");
    copy_to_user(buf, buffer, 1024);
    return count;
}

static struct file_operations vd_fops = {
    .read = test_device_read,
    .write = test_device_write,
    .open = test_device_open,
    .release = test_device_release
};
```

```
};

int __init test_device_init(void) {
    if(register_chrdev(300, "test_device", &vd_fops) < 0 )
        printk(KERN_ALERT "driver init failed\n");
    else
        printk(KERN_ALERT "driver init successful\n");
    buffer = (char*)kmalloc(1024, GFP_KERNEL);
    if(buffer != NULL)
        memset(buffer, 0, 1024);
    return 0;
}

void __exit test_device_exit(void) {
    unregister_chrdev(250, "test_device");
    printk(KERN_ALERT "driver cleanup successful\n");
    kfree(buffer);
}

module_init(test_device_init);
module_exit(test_device_exit);
MODULE_LICENSE("GPL");
```

```
obj-m += test_device.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
```