망고100 보드로 놀아보자 -13

리눅스 디바이스 드라이버 개요

http://cafe.naver.com/embeddedcrazyboys

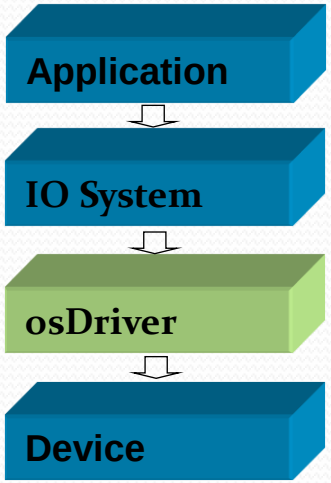
http://www.mangoboard.com

디바이스 드라이버 개요

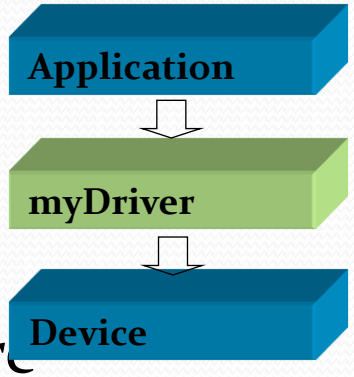
- 디바이스(Device)
  - 네트워크 어댑터, LCD 디스플레이, PCMCIA, Audio, 터미널, 키보드, 하드디스크, 플로피디스크, 프린터 등과 같은 주변 장치들을 말함
  - 디바이스의 구동에 필요한 프로그램, 즉 디바이스 드라이버가 필수적으로 요구됨
- Device Driver
  - 실제 장치 부분을 추상화 시켜 사용자 프로그램이 정형화된 인터페이스를 통해 디바이스를 접근할 수 있도록 해주는 프로그램
  - 디바이스 관리에 필요한 정형화된 인터페이스 구현에 요구되는 함수와 자료구조의 집합체
  - 표준적으로 동일 서비스 제공을 목적으로 커널의 일부분으로 내장
  - 응용프로그램이 H/W를 제어할 수 있도록 인터페이스 제공
  - 하드웨어 독립적인 프로그램을 작성을 가능하게 함

디바이스 드라이버 형태

- Device Driver Interface
  - Standard Device Driver Interface
    - UNIX compatible I/O system interface
      : open(), close(), read(), write(), ioctl()

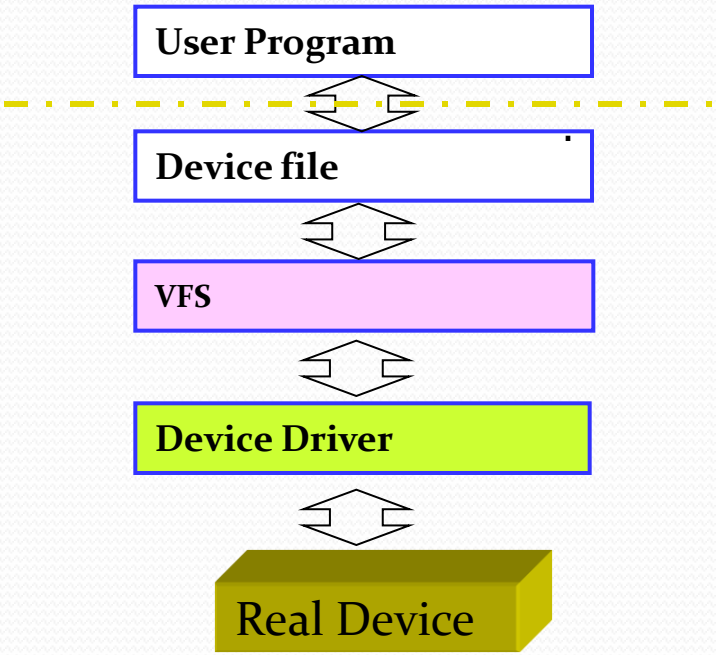


- Non-standard Device Driver Interface
  - Completely user-defined
  - Custom interface
  - May be more appropriate for some hardware



리눅스 디바이스 드라이버

- 사용자 관점에서의 디바이스 드라이버
  - 사용자는 디바이스 자체에 대한 자세한 정보를 알 필요 없음
  - Device는 하나의 파일로 인식됨
  - file에 대한 접근을 통하여 real device에 접근 가능



리눅스 디바이스 드라이버(2)

- 리눅스에서의 디바이스
  - Linux에서 Device는 특별한 하나의 파일처럼 취급되고, access가 가능함.
    - 사용자는 File operation을 적용할 수 있음
  - 각 디바이스는 Major number와 Minor number를 갖음
- Device Driver의 종류
  - 문자 디바이스 드라이버
  - 블록 디바이스 드라이버
  - 네트워크 디바이스 드라이버

Char Device(문자 디바이스)

- 문자 디바이스의 특징
  - 자료의 순차성을 지닌 장치
  - 버퍼 캐쉬를 사용하지 않음
  - 장치의 raw data를 사용자에게 제공
  - Terminal, Serial/Parallel, Keyboard, Sound Card, Scanner, Printer

- 리눅에서의 문자 디바이스

null : black hole
tty* : virtual console
pt* : pseudo-terminal

```
crw--w--w-   0 root   root       5,   1  Oct  1  1998 console
crw-rw-rw-   1 root   root       1,   3  May  6  1998 null
crw-------   1 root   root       4,   0  May  6  1998 tty
crw-rw----   1 root   disk      96,   0  Dec 10  1998 pt0
crw-------   1 root   root       5,  64  May  6  1998 cua0
```

파일 관련 정보 중 첫 문자인 C는 char device를 의미

Block Device(블록 디바이스)

- Block device 특징
  - random access 가능
  - 블록 단위의 입출력이 가능한 장치
  - 버퍼캐쉬에 의한 내부 장치 표현
  - 파일 시스템에 의해 mount 되어 관리되는 장치
  - 디스크, Ram Disk, CD-ROM 등
- 리눅스에서의 Block device

fd* : Floppy disk
Hd* : Hard disk
sda : SCSI disk

```
brw-------   1 root     floppy     2,   0 May  6  1998 fd0
brw-rw----   1 root     disk       3,   0 May  6  1998 hda
brw-rw----   1 root     disk       3,   1 May  6  1998 hda1
brw-rw----   1 root     disk       8,   0 May  6  1998 sda
brw-rw----   1 root     disk       8,   1 May  6  1998 sda1
```

파일 관련 정보 중 첫 문자인 b는 block device를 의미

Network Device(네트워크 디바이스)

- Network device 특징
  - 대응하는 장치파일이 없음
  - 네트워크 통신을 통해 패킷을 송수신할 수 있는 장치
  - 응용프로그램과의 통신은 표준 파일 시스템관련 콜 대신 socket(), bind() 등의 시스템 콜 사용
  - Etherent, PPP, ATM, ISDN 등이 있음

Major & Minor Number

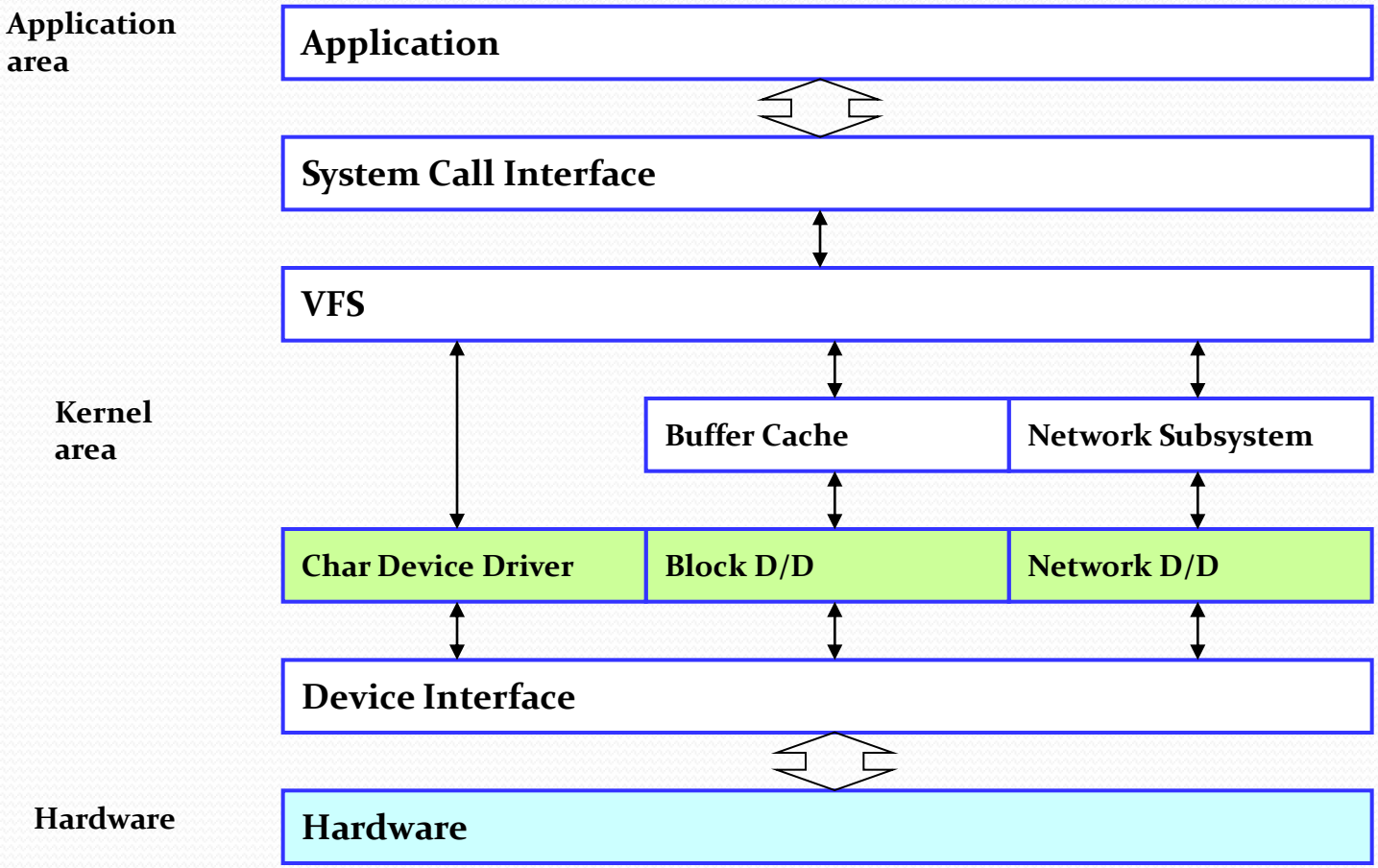
- Major number(주번호)
  - 커널에서 디바이스 드라이버를 구분/연결하는데 사용
  - 같은 Device의 종류를 지칭, 1Byte (0~255사이의 값)
- Minor number(부번호)
  - 디바이스 드라이버 내에서 장치를 구분하기 위해 사용
  - 각 Device의 부가적인 정보를 나타냄, 2Byte (부번호)
  - 하나의 디바이스 드라이버가 여러 개의 디바이스 제어 가능
- $ ls -al /dev/hda*

```
brw-rw----   1 root     disk     1,  0 May  6  1998 hda
brw-rw----   1 root     disk     1,  1 May  6  1998 hda1
brw-rw----   1 root     disk     1,  2 May  6  1998 hda2
brw-rw----   1 root     disk     1,  3 May  6  1998 hda3
```

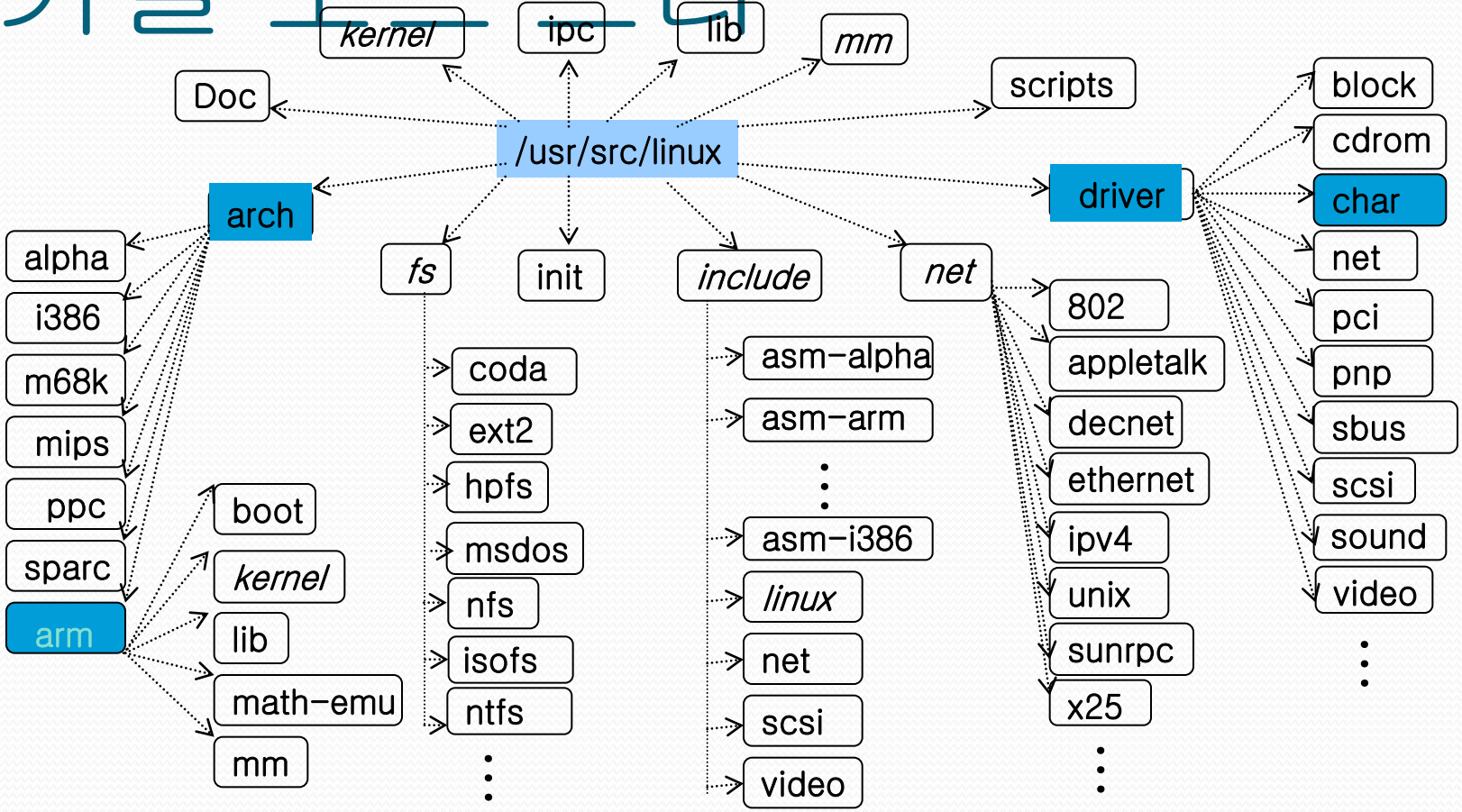
주번호 부번호

디바이스 드라이버 구조

- 리눅스 시스템 구조 상의 디바이스 드라이버



커널 소스 트리


/usr/src/linux
kernel
ipc
lib
mm
Doc
scripts
arch
alpha
i386
m68k
mips
ppc
sparc
arm
boot
kernel
lib
math-emu
mm
fs
coda
ext2
hpfs
msdos
nfs
isofs
ntfs
init
include
asm-alpha
asm-arm
asm-i386
linux
net
scsi
video
net
802
appletalk
decnet
ethernet
ipv4
unix
sunrpc
x25
driver
block
cdrom
char
net
pci
pnp
sbus
scsi
sound
video

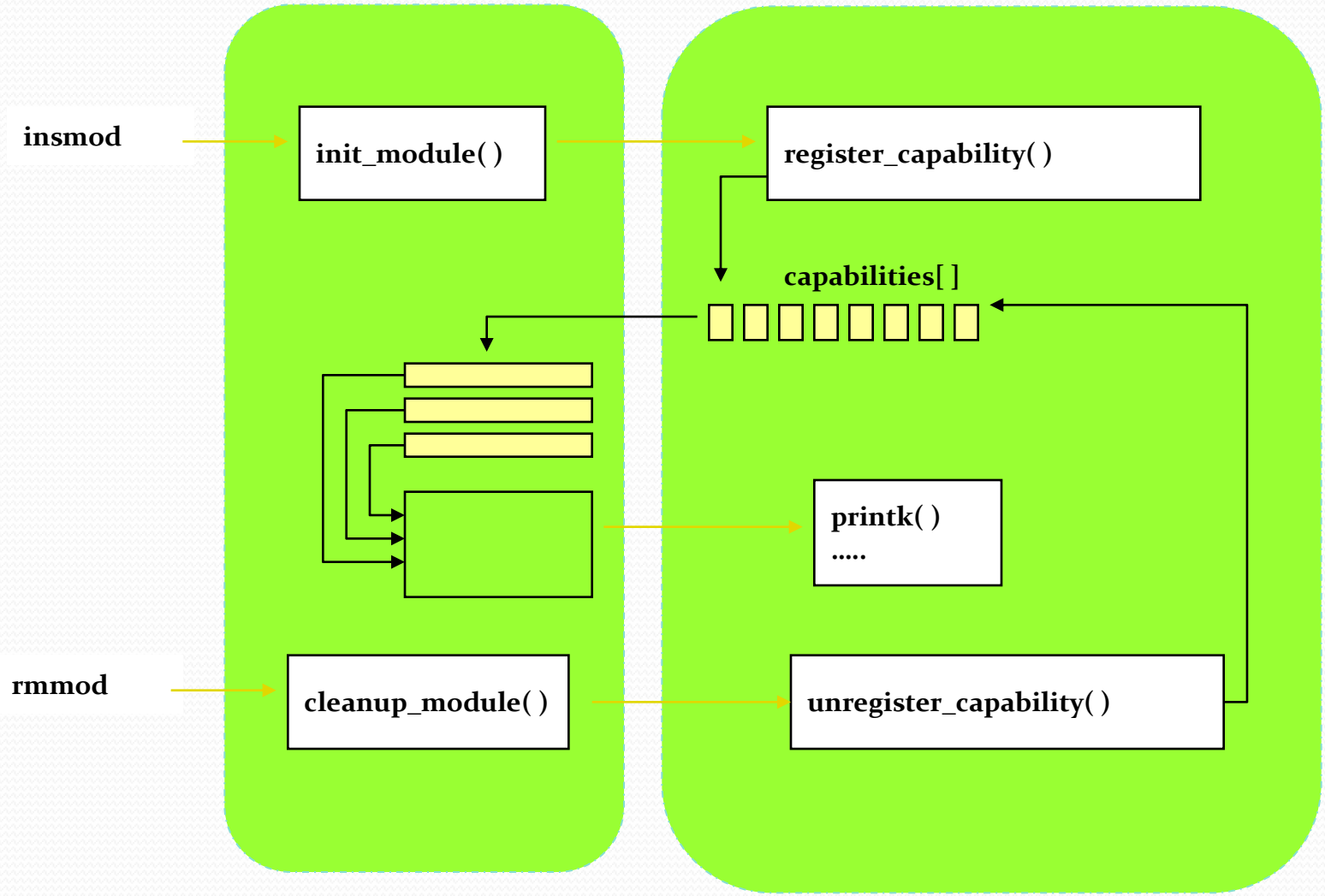
커널 모듈(kernel Module)

- 시스템 부팅 후에 동적으로 loading 할 수 있는 커널 구성요소
- 커널을 다시 컴파일 하거나 시스템 리부팅 할 필요 없이 커널의 일부분을 교체하는 것이 가능
- 디바이스 드라이버, 파일 시스템, 네트워크 프로토콜 등이 모듈로 제공됨
- 컴파일한 커널 버전 정보가 들어가야 하고, 현재 실행되고 있는 커널 버전과 일치해야 함
  - <linux/module.h>에 정의되어 있음
  - 모듈 정보는 전체 모듈에서 하나만 존재해야 함
- 일반 응용 프로그램과의 차이점
  - main() 함수가 없음
  - 커널에 로딩 및 제거 될 때 불러지는 함수가 존재
    - Loading 시 - int init_module(void) 함수 호출
    - Unloading 시 - void cleanup_module() 함수 호출

Linux Device Driver 특성

- Linux device driver의 공통적 특성
  - 커널 코드
    - 디바이스 드라이버는 커널의 한 부분이므로, 커널의 다른 코드와 마찬가지로 잘 못되면 시스템에 치명적인 피해를 줄 수 있다
  - 커널 인터페이스
    - 디바이스 드라이버는 리눅스 커널이나 자신이 속한 서브시스템에 표준 인터페이스를 제공해야 한다.
  - 커널 메커니즘과 서비스
    - 디바이스 드라이버는 메모리 할당, 인터럽트 전달, wait queue같은 표준 커널 서비스를 사용할 수 있다.
  - Loadable
    - 대부분의 리눅스 디바이스 드라이버는 커널 모듈로서, 필요할 때 로드하고 더 이상 필요하지 않을 때 언로드 할 수 있다.
  - 설정가능(Configurable)
    - 리눅스 디바이스 드라이버를 커널에 포함하여 컴파일 할 수 있다. 어떤 장치를 넣을 것인지는 커널을 compile 할 때 설정할 수 있다

커널과 모듈의 링크 개념도



커널 모듈의 작성

- 예제 프로그램
  - 커널에 모듈이 로딩될 때 “hello mango world”를 출력
  - 모듈이 제거될 때 “Good Bye”를 출력
  - Source file : hello.c

```
/* hello.c */

#include <linux/module.h> /* 모든 모듈에 필요 */
#include <linux/kernel.h> /* printk() 등에 필요 */

static  int __init hello_init(void) { // 모듈이 로딩될 때 호출
      printk (“hello mango world\n”);
                  // from a text console, not X-terminal.
      return 0;
      // 0: success , 기타 - fail
}
static void  __exit hello_exit(void)
{
      printk (“KERN_ALERT “Goodbye world”);
}
module_init(hello_init);
Module_exit(hello_exit);
MODULE_LICENCE(“GPL”);
```

커널 모듈의 컴파일(커널에 포함)

- 커널 소스 푼 디렉토리로 이동
- #cp hello.c arch/arm/mach-s5pc100/
- #vi arch/arm/mach-s5pc100/Makefile 수정

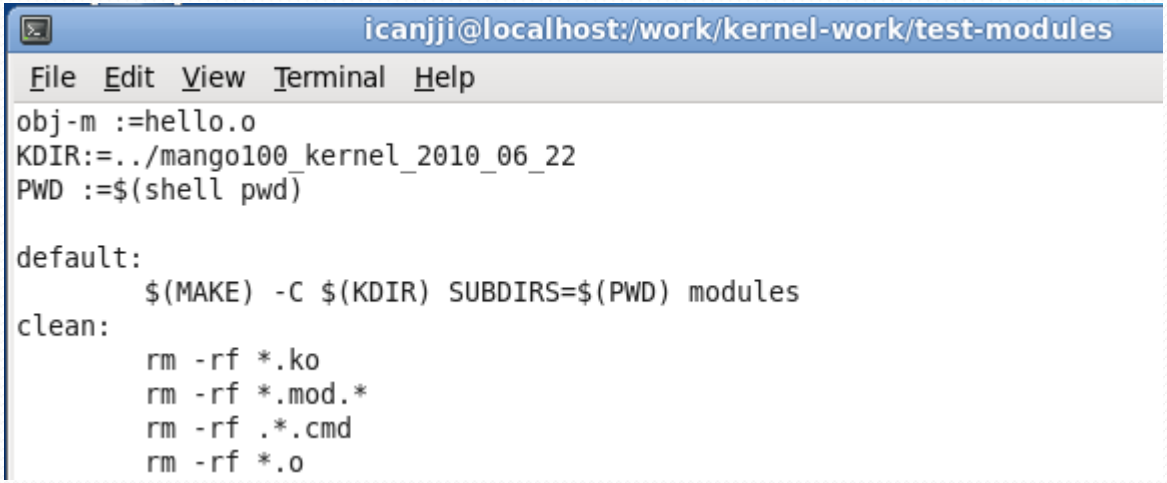
```
# Button EINT suppoty
obj-$(CONFIG_MACH_SMDKC100)     += s5pc100-button.o
obj-m                           +=hello.o
```

- #make modules

```
CC      arch/arm/mach-s5pc100/hello.mod.o
LD [M]  arch/arm/mach-s5pc100/hello.ko
```

커널 모듈의 컴파일(Makefile작성)

- 커널 소스 푼 디렉토리로 이동
- #mkdir test-module
- #cp hello.c test-module
- #vi Makefile



- #make

module 파일시스템에 포함방법

- NFS 로 Mount 하는 방법
- 저장장치를 이용하는 방법
- 파일시스템에 포함하는 방법
- Etc...

module파일시스템에 포함방법 (NFS 로 Mount 하는 방법)

Mango(타겟) board console창에서 입력

#mkdir /mnt/nfs

//#echo "/sbin/mount –t nfs –o nolock $(Serverip):$(nfs 디렉토리) $(dir)

#/sbin/mount -t nfs -o nolock 192.168.0.10:/nfsroot /mnt/nfs

#df

```
root@Mango:/# df
Filesystem           1K-blocks      Used Available Use% Mounted on
ubi0:rootfs             233700    168688     59952  74% /
none                        64        52        12  81% /dev
tmpfs                       64        52        12  81% /dev
tmpfs                   100156       472     99684   0% /var/volatile
tmpfs                   100156         0    100156   0% /dev/shm
tmpfs                   100156         0    100156   0% /media/ram
192.168.0.2:/nfsroot  85084704  16757376  64005216  21% /mnt/nfs
```

cd /mnt/nfs

#insmod hello.ko

#lsmod

#rmmod hello.ko

Host linux PC 명령 순서

#cp hello.ko /nfsroot

#ps –aux | grep nfs

```
[root@localhost mango100_kernel_2010_06_22]# ps -aux | grep nfs
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8,
root      4251  0.0  0.0      0     0 ?        S<   21:20   0:00 [nfsiod]
root      4591  0.0  0.0      0     0 ?        S<   21:54   0:00 [nfsd4]
root      4592  0.0  0.0      0     0 ?        S<   21:54   0:00 [nfsd]
```

#rpm –qa nfs

#yum install nfs*

#mkdir -p /home/nfsroot

#ln –s /home/nfsroot /nfsroot

#vi /etc/exports 에 아래 추가

```
/nfsroot        *(rw,no_root_squash,no_all_squash)
```

#ifconfig etho 192.168.0.10 up

```
root@Mango:/mnt/nfs# insmod hello.ko
Hello Mango world
root@Mango:/mnt/nfs# lsmod
Module                  Size  Used by
hello                   1084  0
root@Mango:/mnt/nfs# rmmod hello.ko
Good bye Mango
```

디바이스 드라이버의 작성방법

- 커널 모듈의 형태로 디바이스 드라이버 함수 작성
  - struct file_operations 정의 및 함수 구현
  - init_module, module_exit 정의 및 함수 구현
- 커널에 디바이스 드라이버 등록
  - register_chrdev(), register_blkdev(), register_netdev()
- 컴파일/로딩
  - Insmod
- Make special file
  - Mknod
- 드라이버를 활용하는 응용프로그램 작성 및 테스트

설정 - 드라이버 적재 및 삭제

- 노드 생성 - 노드(파일)를 통해서 입출력 수행
  - mknod /dev/파일이름 드라이버타입 주번호 부번호
  - 예) mknod /dev/keydd c 125 0
  - 생성 후 속성변경 : chmod ug+w /dev/keydd
- 디바이스 드라이버 적재
  - insmod 드라이버명.ko
  - 예) insmod keydd.ko
- 디바이스 드라이버 삭제
  - rmmod 드라이버명.ko
  - 예)rmmod keydd .ko (주의 .ko 붙지 않아도 됨)
- 드라이버의 적재 여부
  - lsmod

디바이스 드라이버 - Etc.

- Device의 정보를 가지는 File들
  - /proc/devices
    - 현재 System에 장착되어 있는 Device들의 정보
  - ./Documentation/devices.txt
    - 현재 Linux System에서 정의되어 있는 Device들의 Major, Minor Number들에 대한 정보
  - ./include/linux/major.h
    - Major Number를 define한 header

문자형 디바이스 드라이버 골격

#include <linux/kernel.h> #include <linux/module.h> #include <linux/init.h>	Header Files
int device_open(…) { … } int device_release(…) { … } ssize_t device_write(…) { … } ssize_t device_read(…) { … }	Function Prototypes

static struct file_operations device_fops = {
…
ssize_t (*read) (…);
ssize_t (*write) (…);
…
int (*open) (…);
int (*release) (…);
…
 	
};
	File Operation

int init_module(void) { … }	모듈 설치 시 초기화 수행
Void module_exit(void) { … }	모듈 제거 시 반환 작업수행