Embedded System Software

Module
Device Driver

Dept. of Computer Science and Engineering
Sogang University, Seoul, KOREA

실습 시간 확인 받을 것

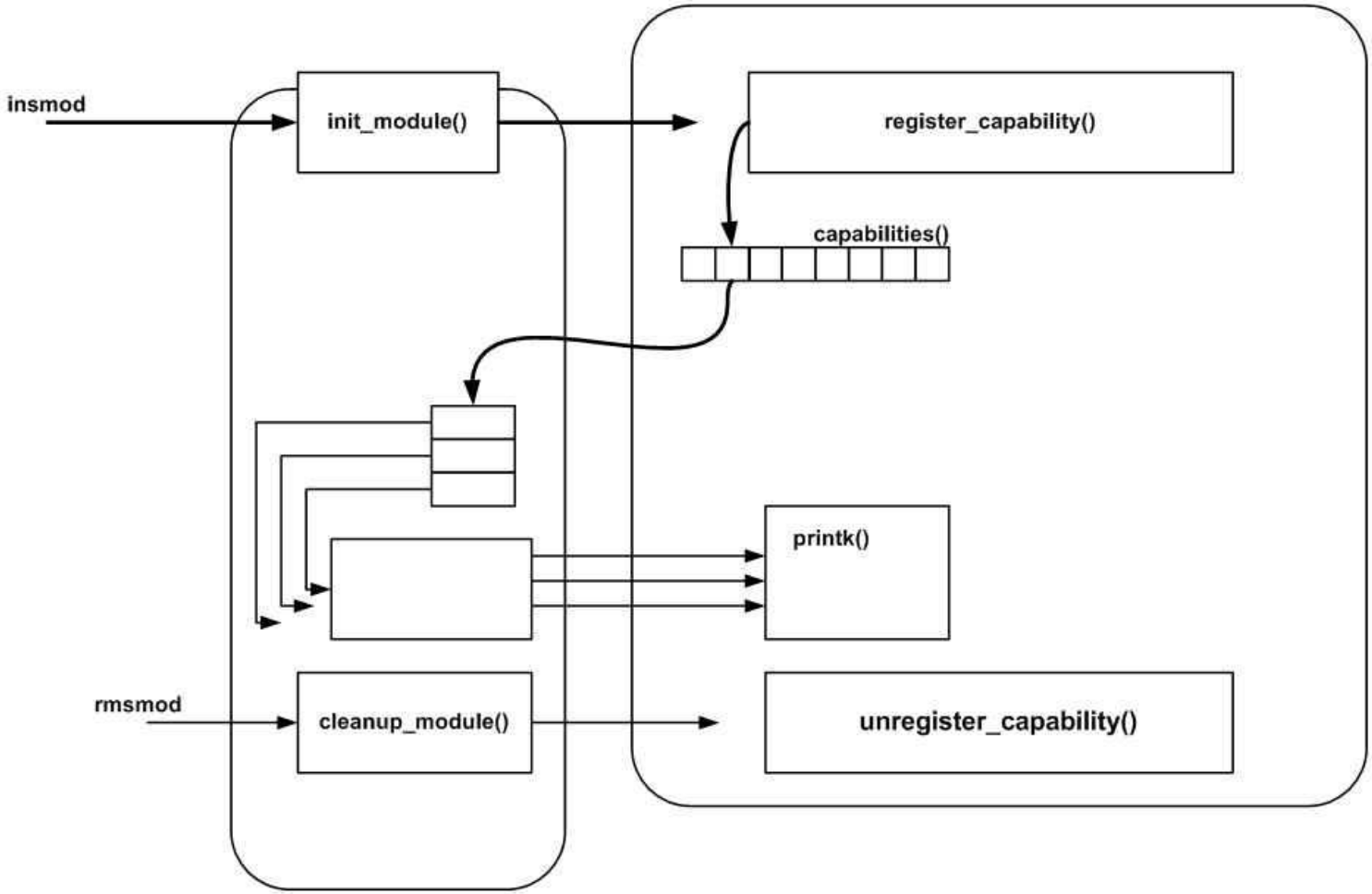
- **Modify module**
  - **module1.c 파일에 mynew_function을 추가하여 입력된 수의 자릿수 (최대 4자리수)와 각 자리수의 숫자를 구하여, module2에서 출력하도록 하는 모듈을 구현**

Module Programming

Module

- 리눅스 시스템이 부팅 된 후에 동적으로 load, unload 할 수 있는 커널의 구성 요소(커널을 다시 컴파일하거나 시스템을 재 부팅 하지 않고도 커널의 일부분을 교체 할 수 있다)
- 모듈의 버전은 현재 실행되고 있는 커널 버전과 같아야 한다(모듈의 버전 정보는 리눅스 커널소스 /include/linux/module.h에 정의 되어 있다)
- main() 함수가 없고, 커널에 loading(적재) 및 unloading(삭제) 할 때 불리는 int init_module(void) 함수와 void cleanup_module() 함수의 선언이 존재한다.

Module Link


insmod
init_module()
register_capability()
capabilities()
printk()
rmsmod
cleanup_module()
unregister_capability()

Module 생성 및 등록

- MODULE_LICENSE()
  - 모듈에 대한 라이센스를 명시하는 커널 매크로
- MODULE_AUTHOR()
  - 모듈의 저작자를 명시하는 커널 매크로
- module_init()
  - 모듈이 시작할때 실행되는 함수
- module_exit()
  - 모듈이 종료 될때 실행되는 함수
- Makefile

```
obj-m := hello_module.o

KDIR := /work/achro4210/kernel
PWD := $(shell pwd)

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

Module 생성 및 등록

- **insmod NAME.ko (등록)**
- **lsmod**
- **rmmod NAME.ko (제거)**

- **모듈 관련 리눅스 명령어**

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

기타 tip

- **Bashrc 다시 적용**
  - **source /root/.bashrc**
- **Tip : printk 출력 확인(보드-minicom에서)**
  - **Echo "7 6 1 7" > /proc/sys/kernel/printk**
- **C crosscompile**
  - **Arm-none-linux-gnueabi-gcc –static –o hello hello.c**
- **전송(host에서)**
  - **Adb push 파일이름 /data/local/tmp**

copy_from_user(), copy_to_user()

- **copy_from_user()**

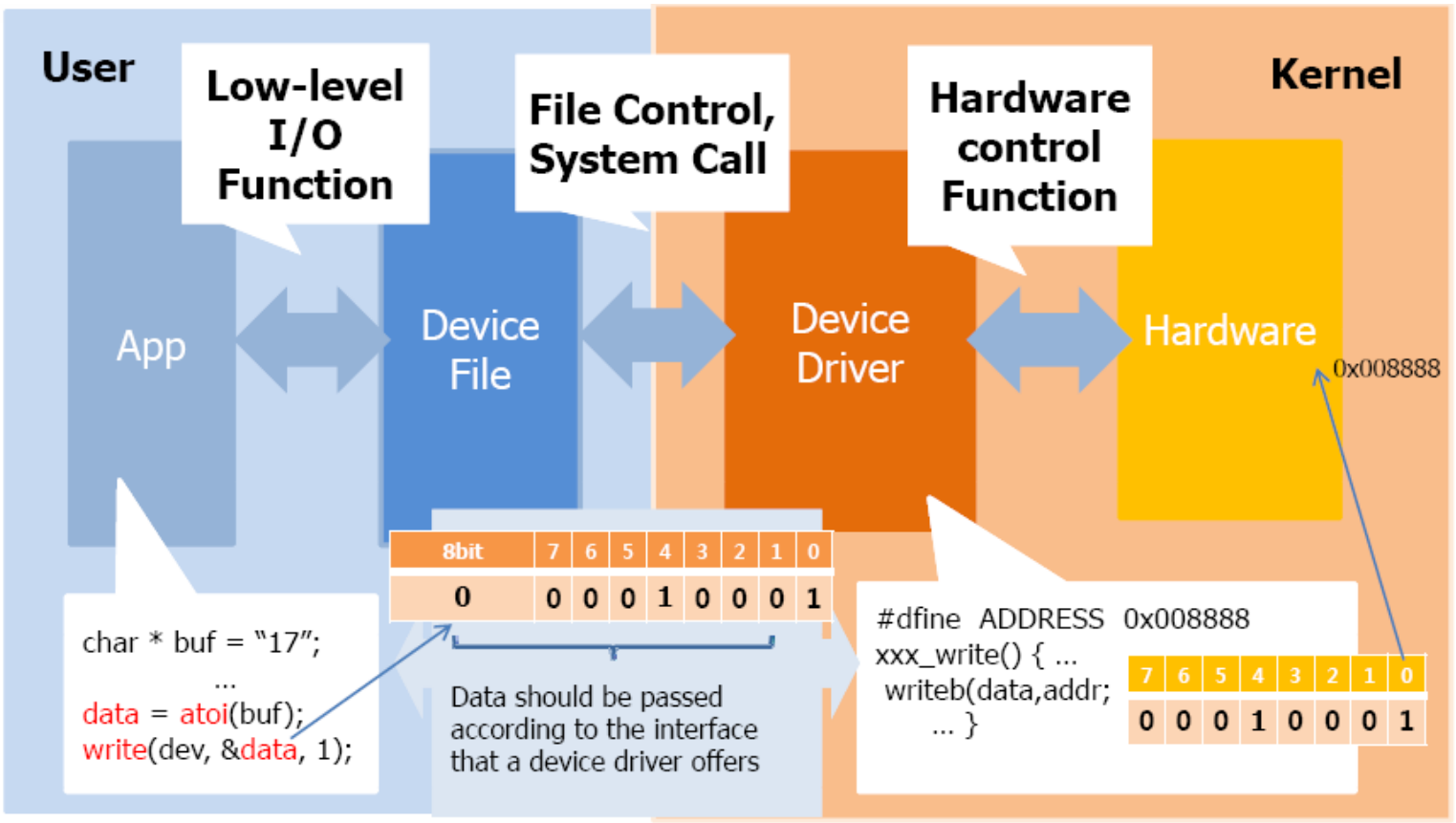
기능	사용자 메모리 블록 데이터를 커널 메모리 블록 데이터에 써넣는다.
형태	#include <asm/uaccess.h> int copy_from_user(void* to, const void __user* from, unsigned long n)
설명	from이 가리키는 주소의 사용자 메모리 블록 데이터를 to가 가리키는 커널 메모리 블록 데이터에 바이트 크기 단위인 n 만큼 써넣는다. 이 함수는 읽어올 공간의 유효성 검사를 수행한다.
매개변수	• from: 사용자 메모리 블록 선두 주소 • to: 커널 메모리 블록 선두 주소 • n: 써넣을 바이트 단위의 크기
반환값	정상적으로 수행이 되었다면 0 이상의 값, 그렇지 않다면 0보다 작은 값을 반환한다.

- **copy_to_user()**

기능	커널 메모리 블록 데이터를 사용자 메모리 블록 데이터에 써넣는다.
형태	#include <asm/uaccess.h> int copy_to_user(void __user* to, const void* from, unsigned long n)
설명	from이 가리키는 주소의 커널 메모리 블록 데이터를 to가 가리키는 사용자 메모리 블록 데이터에 바이트 크기 단위인 n 만큼 써넣는다. 이 함수는 써넣은 공간의 유효성 검사를 수행한다.
매개변수	• from: 커널 메모리 블록 선두 주소 • to: 사용자 메모리 블록 선두 주소 • n: 써넣을 바이트 단위의 크기
반환값	정상적으로 수행이 되었다면 0 이상의 값, 그렇지 않다면 0보다 작은 값을 반환한다.

Device Control Methods

- **Through Device Driver**



Device Control Methods

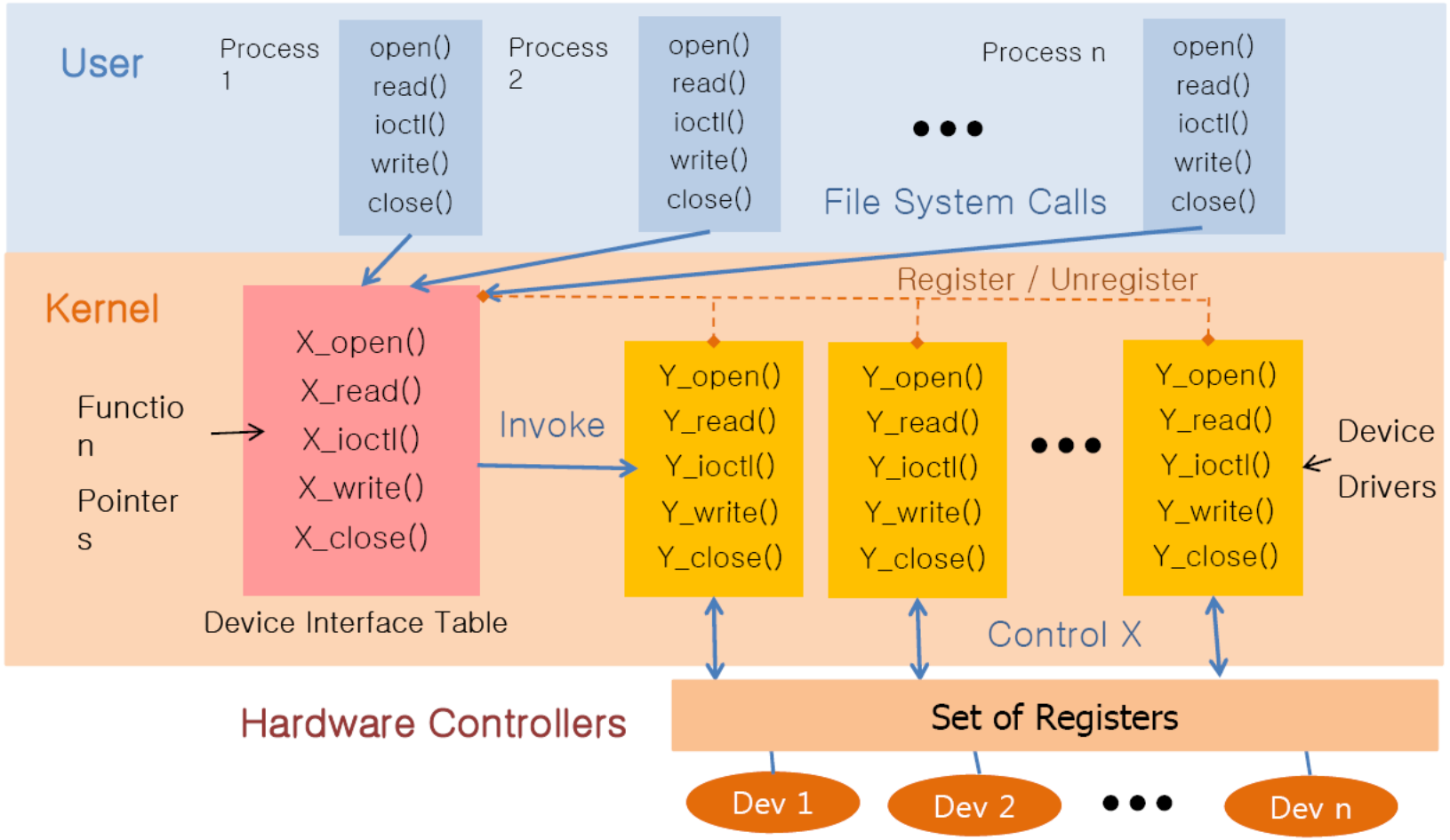
- **Using a Device Driver**
  - **Device driver를 통해 device를 제어하려면, 해당 device에 대한 device file이 필요하다.**
    - /dev : linux system에서 device file들이 위치하는 곳.
  - **mknod *NAME TYPE MAJOR MINOR***
    - device file을 생성해 주는 command
    - NAME : 생성할 device file의 이름을 지정 (/dev/xxx 와 같이 절대경로를 지정한다).
    - TYPE : b (block device) / c (character device)
    - MAJOR : device driver의 major number
    - MINOR : device driver의 minor number (보통, 0)
    - 일반 파일과 같이 rm명령으로 삭제할 수 있다.
  - **Module로 작성된 device driver를 사용하기 위해서는, 현재 실행 중인 kernel에 driver module을 삽입해야 한다.**
  - **insmod *NAME***
    - device driver module을 kernel에 삽입해 주는 command
    - NAME : driver module file의 이름
    - 모듈을 제거할 때는 rmmod 를 사용한다.

Driver 사용 예시(led)

- **insmod fpga_led_driver.ko**
- **mknod /dev/fpga_led c 260 0**
- (mknod /dev/[devicename] [type] [major] [minor])

Device Driver

Device Driver Overview



I/O mapped vs Memory mapped I/O

- **I/O mapped I/O**
  - **I/O 명령을 위한 instruction이 따로 존재한다. CPU의 extra I/O pin 또는 I/O 전용으로 할당된 별도의 bus를 이용하여 I/O 명령을 처리한다.**
  - **I/O 에 접근하기 위한 address space가 memory접근을 위한 address space와는 별도로 존재한다.**
  - **주로 Intel 계열의 processor에서 사용하는 방식이다.**

- **Memory mapped I/O**
  - **memory 에 접근하기 위한 address space와 I/O에 접근하기 위한 address space가 같다.**
  - **memory의 특정 영역이 I/O를 위한 주소로 미리 예약되어 있다.**
  - **주로 68 계열이나 ARM 계열의 processor에서 사용하는 방식이다.**

I/O functions

- **Memory mapping function**
  - 특정 물리 주소 공간을 커널 주소 공간으로 **mapping** 하는 **function**

- 경쟁 처리 함수
  - 동일한 **I/O** 영역(**port or memory**)을 다른 목적으로 사용하는 **driver**들이 동시에 경쟁할 경우, 충돌이 생기지 않도록 영역의 사용 권한을 허가 또는 제한하는 함수

- **I/O** 처리 함수
  - 다른 **architecture**와의 호환성을 위하여 일반적인 **pointer** 연산을 사용하지 않고, **I/O**를 위하여 정의된 **macro** 함수들을 사용한다.

Memory mapping function

```
#include <asm-generic/io.h>
void *ioremap(unsigned long addr, unsigned long
    size);
void *iounmap(unsigned long addr, unsigned long
    size);
```

- **ioremap()**
  - 물리 주소 공간을 커널 주소 공간으로 **mapping**한다.
- **iounmap()**
  - **mapping**한 커널 주소 공간을 해제한다.

Using I/O port – 경쟁 처리 함수

```
#include <linux/ioport.h>
int check_region(unsigned long start, unsigned long len);
struct resource * request_region(unsigned long start,
    unsigned long len, char * name);
void release_region(unsigned long start, unsigned long
    len);
```

- **check_region : 등록할 수 있는 I/O 영역인지 확인한다.**
  (가능하지 않을 경우, 0보다 작은 값 –EBUSY를 반환한다.)
- **request_region: I/O 영역을 등록한다.**
- **release_region: 등록된 I/O 영역을 해제한다.**

Using I/O port – I/O 처리 함수

```
#include <asm-generic/io.h>
unsigned char inb(unsigned short port);
unsigned short inw(unsigned short port);
unsigned long inl(unsigned short port);
void outb(unsigned char data, unsigned short port);
void outw(unsigned short data, unsigned short port);
void outl(unsigned long data, unsigned short port);
void insb(unsigned short port, void *addr, unsigned long
    count);
void outsb(unsigned short port, void *addr, unsigned long
    count);
    ...
```

함수의 구현방식은 리눅스 커널을 사용하려는 프로세서, 아키텍처에 따라 다르게 구현 되어있다.

b(byte), w(word), long(long)

Using I/O memory – 경쟁 처리 함수

```
#include <linux/ioport.h>
int check_mem_region(unsigned long start, unsigned
    long len);
struct resource *request_mem_region(unsigned long
    start, unsigned long len, char * name);
void release_mem_region(unsigned long start, unsigned
    long len);
```

- check_mem_region : 등록할 수 있는 I/O Memory 영역인지 확인한다.
- request_mem_region: I/O memory 영역을 등록한다.
- release_mem_region: 등록된 I/O memory 영역을 해제한다.

Using I/O memory – I/O 처리 함수

```
#include <asm-generic/io.h>
unsigned char readb(unsigned int addr);
unsigned short readw(unsigned int addr);
unsigned long readl(unsigned int addr);
void writeb(unsigned char data, unsigned short addr);
void writew(unsigned short data, unsigned short addr);
void writel(unsigned long data, unsigned short addr);
void readsb(unsigned short addr, void *addr, unsigned long
    count);
void writesb(unsigned short addr, void *addr, unsigned long
    count);
```

함수의 구현방식은 리눅스 커널을 사용하려는 프로세서, 아키텍처에 따라 다르게 구현 되어있다.

b(byte), w(word), long(long)

FND Device Driver

- **Registering / Unregistering a Device**
  - **#include <linux/fs.h>**
  - **register_chrdev(unsigned int major, const char *name, struct file_operations *fops);**
  - **unregister_chrdev(unsigned int major, const char *name);**
- **Assigning File Operations**

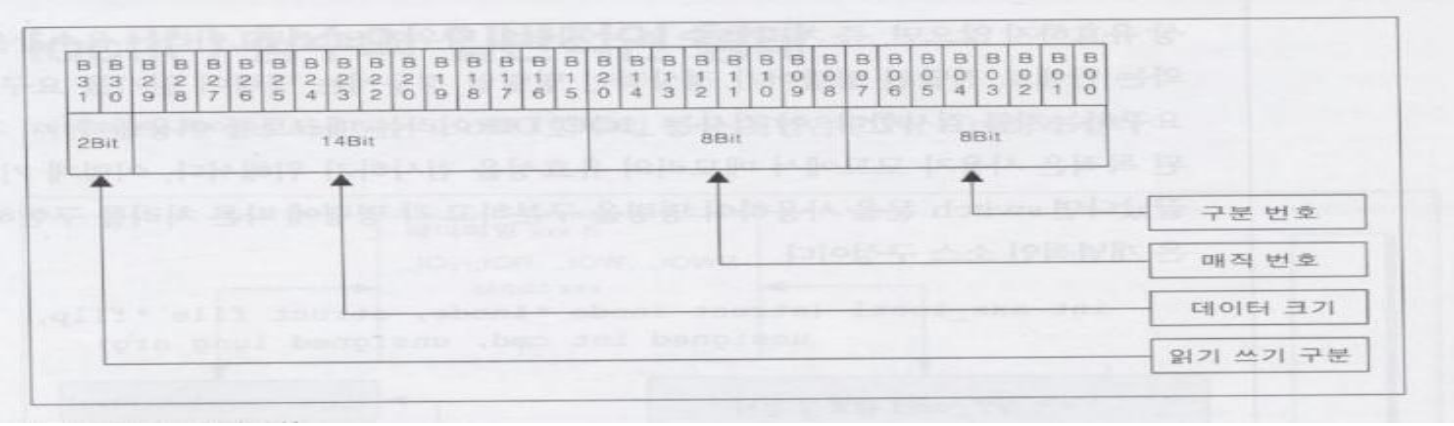
```
struct file_operations xxx_fops {
        .owner    = THIS_MODULE;
        .write    = …;
        . …       = …;
        …
};
```

ioctrl

- **ioctl()**

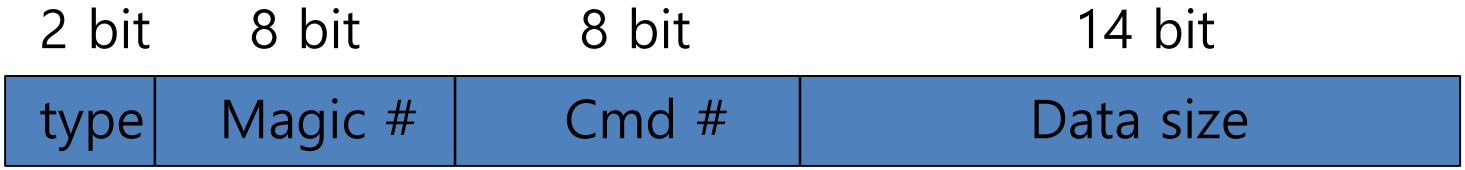
```
ret = ioctl( int fd , int request , char *argp )

int xxx_ioctl(struct inode, struct file *filp, unsigned int cmd, unsigned long arg)
{
        return ret;
}
```

- **Command structure**



ioctl commands

- **Command is an integer (32 bit)**

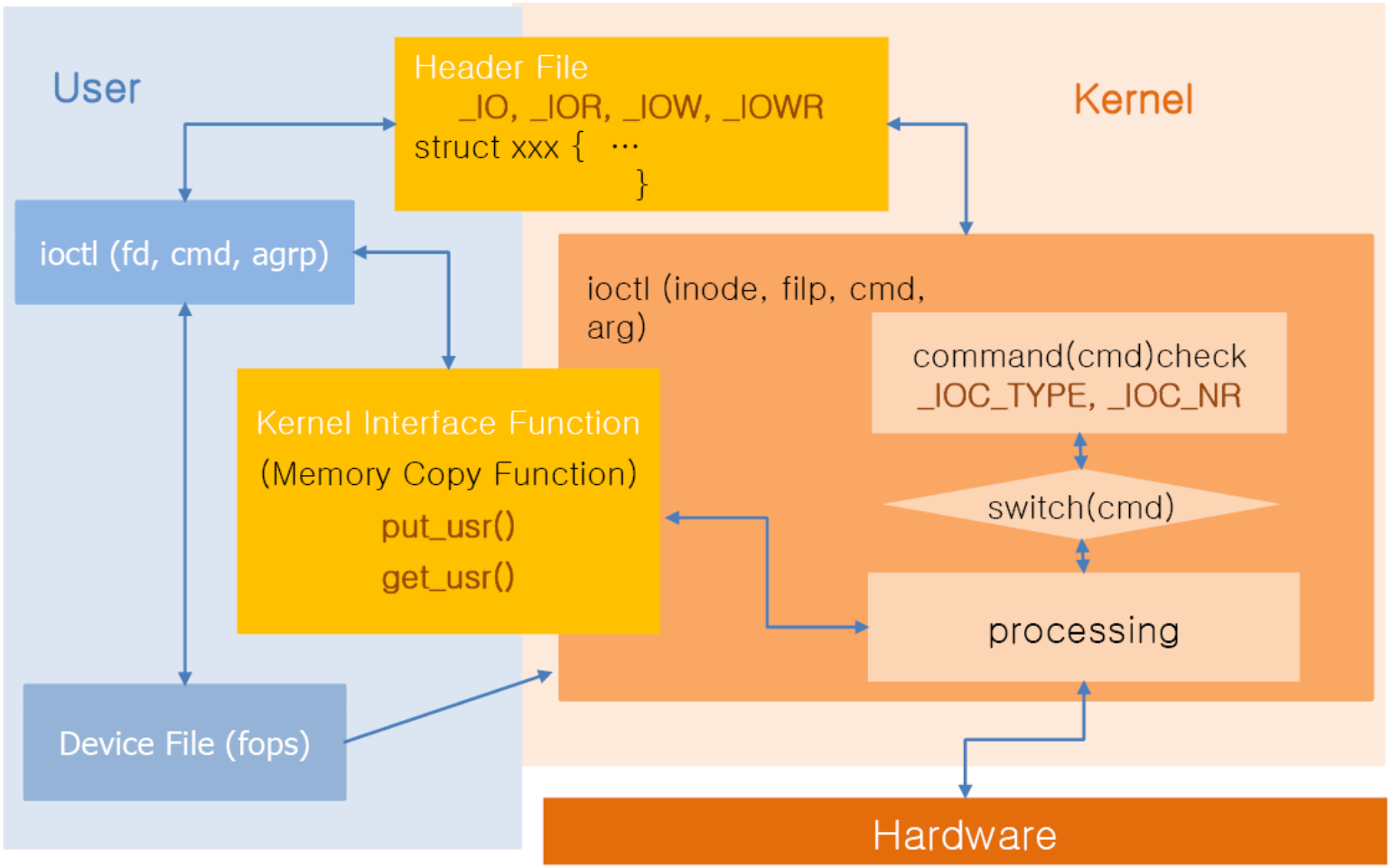


- **Example**
  - **User가 ioctl(fd , 2151694592,....) 를 호출했을 때, 그 command의 의미는**
  - **2151694592 = 10 / 00000001 / 00000001 / 0000010000000 (2진수)**
  - **Type = 2, magic # = 1, cmd # = 1, data size = 256**
- **Command를 만드는 과정이 매우 험난하다 => macro 제공**

ioctl commands

- **Macros to encode / decode a command**
  - **#include <asm/ioctl.h>**
  - **Encoding**
    - _IO(type, number)
    - _IOR(type, number, datatype)
    - _IOW(type, number, datatype)
    - _IOWR(type, number, datatype)
  - **Decoding**
    - _IOC_TYPE(command)
    - _IOC_NR(command)
    - _IOC_DIR(command)
    - _IOC_SIZE(command)

Using ioctl


User
Header File
_IO, _IOR, _IOW, _IOWR
struct xxx { …
}
Kernel
ioctl (fd, cmd, agrp)
ioctl (inode, filp, cmd, arg)
command(cmd)check
_IOC_TYPE, _IOC_NR
Kernel Interface Function
(Memory Copy Function)
put_usr()
get_usr()
switch(cmd)
processing
Device File (fops)
Hardware

ioctl commands

- **Macros to encode / decode a command**
  - **#include <asm/ioctl.h>**
  - **Encoding**
    - _IO(type, number)
    - _IOR(type, number, datatype)
    - _IOW(type, number, datatype)
    - _IOWR(type, number, datatype)
  - **Decoding**
    - _IOC_TYPE(command)
    - _IOC_NR(command)
    - _IOC_DIR(command)
    - _IOC_SIZE(command)