TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 하성용
accept0108@naver.com

37 일차

36 일에 디바이스드라이버 복습

cat /proc/devices // 현재 시스템에서 실제 사용중인 주 번호 확인 명령어

***가상 디바이스 드라이버 구현**

Makefile

```
obj-m :=chr_test.o

KERNEL_DIR := /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

default:
	$(MAKE) -C $(KERNEL_DIR) SUBDIRS=$(PWD) modules
clean:
	$(MAKE) -C $(KERNEL_DIR) SUBDIRS=$(PWD) clean
```

chr_test.c

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/slab.h>
#include <linux/fs.h>
#include <linux/cdev.h>
#include <linux/device.h>
#include <linux/uaccess.h>

#define DEVICE_NAME "mydrv"
#define MYDRV_MAX_LENGTH 4096
#define MIN(a,b) (((a) < (b)) ? (a):(b))

struct class *myclass;
struct cdev *mycdev;
struct device *mydevice;
dev_t mydev;

static char *mydrv_data;
static int mydrv_read_offset, mydrv_write_offset;

static int mydrv_open(struct inode *inode, struct file *file)
{
	printk("%s\n",__FUNCTION__);
	return 0;
}

static int mydrv_release(struct inode * inode, struct file *file)
{
	printk("%s\n",__FUNCTION__);
	return 0;
}

static ssize_t mydrv_read(struct file *file, char *buf, size_t count, loff_t
		*ppos)
{
	if( (buf == NULL) || (count < 0 ) )
		return -EINVAL;
```

```
	if( (mydrv_write_offset - mydrv_read_offset) <= 0 )
		return 0;
	count = MIN( (mydrv_write_offset - mydrv_read_offset), count );
	if( copy_to_user(buf, mydrv_data + mydrv_read_offset, count) );
		return -EFAULT;
	mydrv_read_offset += count;
	return count;
}

static ssize_t mydrv_write(struct file *file, const char *buf, size_t count,
		loff_t *ppos)
{
	if( (buf==NULL) || (count<0) )
		return -EINVAL;
	if( count+mydrv_write_offset >= MYDRV_MAX_LENGTH) {
		/* driver space is too small */
		return 0;
	}
	if( copy_from_user(mydrv_data + mydrv_write_offset, buf, count) )
		return -EFAULT;
	mydrv_write_offset += count;
	return count;
}

struct file_operations mydrv_fops = {
	.owner = THIS_MODULE,
	.read = mydrv_read,
	.write = mydrv_write,
	.open = mydrv_open,
	.release = mydrv_release,
};
int mydrv_init(void)
{
	if( alloc_chrdev_region(&mydev, 0, 1, DEVICE_NAME) < 0 ) {
		return -EBUSY;
	}

	myclass = class_create(THIS_MODULE, "mycharclass");
	if (IS_ERR(myclass)){
		unregister_chrdev_region(mydev, 1);
		return PTR_ERR(myclass);
	}

	mydevice = device_create(myclass, NULL, mydev, NULL, "mydevicefile");
	if(IS_ERR(mydevice)){
		class_destroy(myclass);
		unregister_chrdev_region(mydev, 1);
		return PTR_ERR(mydevice);
	}

	mycdev = cdev_alloc();
	mycdev -> ops = &mydrv_fops;
	mycdev -> owner = THIS_MODULE;
	if ( cdev_add(mycdev, mydev, 1) < 0 ) {
		device_destroy(myclass, mydev);
		class_destroy(myclass);
		unregister_chrdev_region(mydev, 1);
		return -EBUSY;
	}
```

```
	mydrv_data = (char *) kmalloc(MYDRV_MAX_LENGTH * sizeof(char),
GFP_KERNEL);
	mydrv_read_offset = mydrv_write_offset = 0;
	return 0;
}

void mydrv_cleanup(void)
{
	kfree(mydrv_data);
	cdev_del(mycdev);
	device_destroy(myclass, mydev);
	class_destroy(myclass);
	unregister_chrdev_region(mydev, 1);
}

module_init(mydrv_init);
module_exit(mydrv_cleanup);
MODULE_LICENSE("GPL");
```

// make 실행

```
yong@yong-Z20NH-AS51B5U:~/driver1$ make
make -C /lib/modules/4.13.0-38-generic/build SUBDIRS=/home/yong/driver1 modules
make[1]: Entering directory '/usr/src/linux-headers-4.13.0-38-generic'
  CC [M]  /home/yong/driver1/chr_test.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/yong/driver1/chr_test.mod.o
  LD [M]  /home/yong/driver1/chr_test.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.13.0-38-generic'
```

// ls

```
yong@yong-Z20NH-AS51B5U:~/driver1$ ls
chr_test.c   chr_test.mod.c  chr_test.o  modules.order
chr_test.ko  chr_test.mod.o  Makefile    Module.symvers
```

sudo → 관리자권한 // sudo insmod chr_test.ko //나는 관리자권한으로 이장치를 연결시키겠다

```
yong@yong-Z20NH-AS51B5U:~/driver1$ sudo insmod chr_test.ko
[sudo] password for yong:
```

// 확인

```
yong@yong-Z20NH-AS51B5U:~/driver1$ ls -l /dev/mydevicefile
crw------- 1 root root 243, 0  4월 13 13:07 /dev/mydevicefile
```

mydrv_test.c

```
#include <stdio.h>
#include <fcntl.h>

#define MAX_BUFFER		26
char buf_in[MAX_BUFFER];
char buf_out[MAX_BUFFER];

int main(void)
{
	int fd, i, c = 65;
	if( (fd = open("/dev/mydevicefile", O_RDWR)) < 0) {
		perror("open error");
		return -1;
	}
	for(i=0; i<MAX_BUFFER; i++){
		buf_out[i] = c++;
```

```
                buf_in[i] = 65;
        }

        for(i=0; i<MAX_BUFFER; i++){
                fprintf(stderr, "%c", buf_in[i]);
        }
        fprintf(stderr, "\n");

        write(fd, buf_out, MAX_BUFFER);
        read(fd, buf_in, MAX_BUFFER);

        for(i=0; i<MAX_BUFFER; i++){
                fprintf(stderr, "%c", buf_in[i]);
        }
        fprintf(stderr, "\n");

        close(fd);
        return 0;
}
```

// gcc mydrv_test.c

```
yong@yong-Z20NH-AS51B5U:~/driver1$ gcc mydrv_test.c
mydrv_test.c: In function ‘main’:
mydrv_test.c:25:2: warning: implicit declaration of function ‘write’ [-Wimplicit-function-declaration]
  write(fd, buf_out, MAX_BUFFER);
  ^
mydrv_test.c:26:2: warning: implicit declaration of function ‘read’ [-Wimplicit-function-declaration]
  read(fd, buf_in, MAX_BUFFER);
  ^
mydrv_test.c:33:2: warning: implicit declaration of function ‘close’ [-Wimplicit-function-declaration]
  close(fd);
```

// permission denied 란? 권한이 없어서 실행을 못한다

```
yong@yong-Z20NH-AS51B5U:~/driver1$ ./a.out
open error: Permission denied
```

// sudo 로 관리자권한을 가져와서 실행

```
yong@yong-Z20NH-AS51B5U:~/driver1$ sudo ./a.out
[sudo] password for yong:
AAAAAAAAAAAAAAAAAAAAAAAAA
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```