

14주차 3차시 디바이스 드라이버 구현

【학습목표】

1. 디바이스 드라이버 파일 생성 및 작성을 설명할 수 있다.
2. 디바이스 드라이버를 구현할 수 있다.

학습내용1 : 디바이스 드라이버 모듈

1. 디바이스 드라이버 구성

* 디바이스 처리

- 응용 프로그램은 디바이스 파일을 통해 디바이스 드라이버 함수와 연결
- 디바이스 드라이버 : 하드웨어를 제어
- 디바이스 파일 : 디바이스 형식과 주번호를 이용해 커널내에 등록된 디바이스 드라이버 함수를 연결
- 문자 디바이스 드라이버 : 응용 프로그램에서 해당 디바이스 드라이버와 디바이스 파일을 호출
- 블록/네트워크 디바이스 드라이버 : 커널에서 직접 호출해서 사용

2. 문자 디바이스 드라이버의 기본 구성

* 문자 디바이스 드라이버의 제작 최소 항목

- 커널 소스 헤더파일
- 저소준 파일 입출력에 대응하는 file_operations 구조체에 등록할 함수
- file_operations 변수
- 문자 디바이스 드라이버를 등록하는 모듈 초기화 함수
- 문자 디바이스 드라이버를 제거하는 모듈 마무리 함수

학습내용2 : 디바이스 드라이버파일 생성

```

01 #include <linux/module.h>
02 #include <linux/kernel.h>
03 #include <linux/fs.h>
04 #include <linux/init.h>
05
06 #define TEST_DEV_NAME    "test_dev"
07 // 디바이스 파일 이름
08 #define TEST_DEV_MAJOR    240
09 // 디바이스 파일의 주번호
10
11 int test_open(struct inode *inode,
12 struct file *filp)
13 {
14     ..... (open 처리) .....
15     return 0;
16 }
17
18 int test_release(struct inode *inode,
19 struct file *filp)
20 {
21     ..... (close 처리) .....
22     return 0;
23 }
24
25 struct file_operations test_fops =
26 {
27     .owner    = THIS_MODULE,
28     .open     = test_open,
29     .release  = test_release,
30 };
31
32 int test_init(void)
33 // 디바이스를 커널에 모듈로 적재시 수행되는 함수
34 {
35     register_chrdev(TEST_DEV_MAJOR,
36 TEST_DEV_NAME, &test_fops);
37     ..... (초기화 처리) .....
38     return 0;
39 }
40
41 void test_exit(void)
42 // 커널에서 디바이스를 제거할 때 수행되는 함수
43 {
44     ..... (마무리 처리) .....
45     unregister_chrdev(TEST_DEV_MAJOR,
46 TEST_DEV_NAME);
47 }
48
49 MODULE_LICENSE("GPL");
50 module_init(test_init);
51 module_exit(test_exit);

```

1. 디바이스 드라이버 구현

디바이스 드라이버 구현 및 테스트 과정



* 모듈

- 목적 코드로 컴파일
- 실행중인 커널에 insmod 명령에 의해 적재
- Insmod 유틸리티 : 실행중인 커널 심볼 테이블을 사용해 모듈 내의 미해결 심볼을 적절한 주소로 연결

2. 디바이스 드라이버 구현 예

① 디바이스 파일 생성

- 가상 디바이스 드라이버 구현

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

#define TEST_DEV_NAME    "test_dev"
#define TEST_DEV_MAJOR   240

int test_open(struct inode * inode, struct file * filp)
{
    //부번호 number에 저장
    int number = MINOR(inode->i_rdev);
    printk("Character Devie Open : Miner Number is
           %d\\n", number);
    return 0;
}

int test_reslease(struct inode * inode, struct file * filp)
{
    .....(close 처리).....
    return 0;
}

struct file_operations test_fops =
{
    .owner = THIS_MODULE,
    .open = test_open,
    .ioctl = test_ioctl,
    .write = test_write,
    .read = test_read,
    .release = test_reslease,
};

int test_init(void)
{
    register_chrdev(TEST_DEV_MAJOR, TEST_DEV_NAME, &test_fops);
    .....(초기화 처리).....
    return 0;
}
```

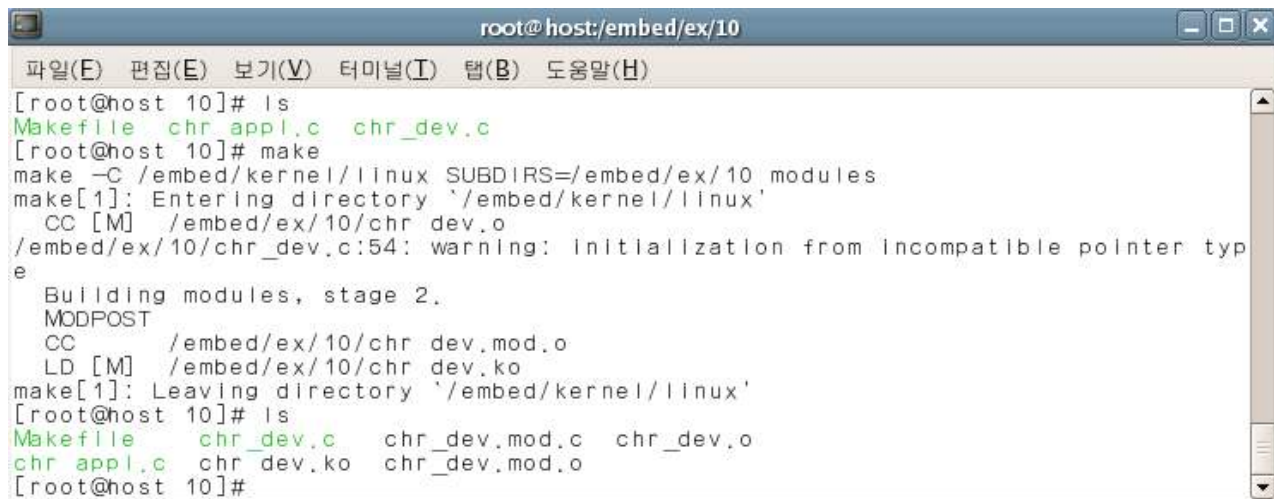
```

void test_exit(void)
{
    .....(마무리처리).....
    unregister_chrdev(TEST_DEV_MAJOR, TEST_DEV_NAME);
}

MODULE_LICENSE("GPL");
module_init(test_init);
module_exit(test_exit);

```

② Make 유틸리티를 실행해 모듈 생성



```

root@host:/embed/ex/10
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@host 10]# ls
Makefile chr_appl.c chr_dev.c
[root@host 10]# make
make -C /embed/kernel/linux SUBDIRS=/embed/ex/10 modules
make[1]: Entering directory '/embed/kernel/linux'
  CC [M]  /embed/ex/10/chr_dev.o
/embed/ex/10/chr_dev.c:54: warning: initialization from incompatible pointer typ
e
Building modules, stage 2.
MODPOST
  CC      /embed/ex/10/chr_dev.mod.o
  LD [M]  /embed/ex/10/chr_dev.ko
make[1]: Leaving directory '/embed/kernel/linux'
[root@host 10]# ls
Makefile  chr_dev.c  chr_dev.mod.c  chr_dev.o
chr_appl.c  chr_dev.ko  chr_dev.mod.o
[root@host 10]#

```

③ 가상 문자 디바이스를 사용하는 응용프로그램 작성(chr_appl.c)

```

01 #include <stdio.h>
02 #include <sys/types.h>
03 #include <sys/stat.h>
04 #include <fcntl.h>
05 #include <sys/ioctl.h>
06 #include <unistd.h>
07
08 #define DEVICE_FILE_NAME "/dev/chr_dev" // 디바이스 파일
09
10 int main(int argc, char *argv[]) // argv 값을 받아 디바이스
11 { // 파일의 IOCTL cmd 값으로 사용
12     int device;
13     char wbuf[128] = "Write buffer data";
14     char rbuf[128] = "Read buffer data";
15     int n = atoi(argv[1]);
16
17     device = open(DEVICE_FILE_NAME, O_RDWR|O_NDELAY);
18     if( device >= 0 ) {
19         printf("Device file Open\n");
20         ioctl(device, n); // argv 값을 디바이스 파일에 cmd 값으로 전달
21         write(device, wbuf, 10); // wbuf 값을 디바이스 파일에 전달
22         printf("Write value is %s\n", wbuf);
23         read(device, rbuf, 10);
24         printf("read value is %s\n", rbuf);
25     }
26     else
27         perror("Device file open fail");
28
29     return 0;
30 }

```

④ 응용프로그램을 타겟 시스템용으로 교차 컴파일



```

root@host:embed/ex/10
파일(E) 편집(E) 보기(V) 터미널(I) 탭(B) 도움말(H)
[root@host 10]# arm-linux-gcc -o chr_appl chr_appl.c
[root@host 10]# ls
Makefile  chr_appl.c  chr_dev.ko  chr_dev.mod.o
chr_appl  chr_dev.c  chr_dev.mod.c  chr_dev.o
[root@host 10]#

```

- ⑤ 디바이스 드라이버 모듈 chr_dev.ko 파일과 컴파일한 응용 프로그램 chr_appl을 타겟 시스템으로 전송
 ⑥ 생성된 디바이스 드라이버 모듈을 적재하고 적재 여부 확인

```

root@host:~
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@xhyper ~]# ls
chr appl      hello-arm    nosymbol.ko  symbol2.ko
chr_dev.ko    helloProc.ko runX hybus.sh
gettaskinfo.ko mb link.sh   symbol1.ko
[root@xhyper ~]# insmod chr_dev.ko
Using chr_dev.ko
Registration Character Device to Kernel
Major Number:0
[root@xhyper ~]# lsmod
Module                Size  Used by      Tainted: P
chr_dev 1856 0 - Live 0xbf01e000
hcd_otg242 28420 0 - Live 0xbf016000
CC2420 6140 0 - Live 0xbf013000
st_motor 3016 0 - Live 0xbf011000
led 1804 0 - Live 0xbf00f000
key 2700 0 - Live 0xbf00d000
fnd 3176 0 - Live 0xbf00b000
dot 2228 0 - Live 0xbf009000
dc_motor 3572 0 - Live 0xbf007000
clcd 2580 0 - Live 0xbf005000
Ide_disk 13856 0 - Live 0xbf000000
[root@xhyper ~]#

```

- 디바이스 드라이버 모듈을 커널에 적재하고 출력되는 메시지를 관찰한다.
- 적재된 디바이스 드라이버를 lsmod 명령으로 확인한다.

- ⑦ 응용 프로그램과 디바이스 드라이버를 연결시킬 디바이스 파일을 만들고 확인

```

root@host:~
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@xhyper ~]# mknod /dev/chr_dev c 240 0
[root@xhyper ~]# ls -l /dev/chr_dev
crw-r--r-- 1 root root 240, 0 Jun 13 17:01 /dev/chr_dev
[root@xhyper ~]#

```

- ⑧ 명령행 인자를 사용해 응용 프로그램을 실행→디바이스 드라이버 모듈 제거

```

root@host:~
파일(E) 편집(E) 보기(V) 터미널(T) 탭(B) 도움말(H)
[root@xhyper ~]# ./chr_appl 1
Virtual Character Device Open: Minor Number is 0
Device file Open
cmd value is 1
write data: Write buffer data
Write value is Write buffer data
read data: Read buffer data
read value is Read buffer data
Virtual Character Device Release
[root@xhyper ~]# rmmod chr_dev
Unregistration Character Device to Kernel
[root@xhyper ~]# █

```

- 응용 프로그램을 실행한다. 출력되는 부번호와 디바이스 파일의 부번호가 모두 0으로 일치함을 확인한다. 출력 메시지를 확인하고 디바이스 모듈의 동작과정을 살펴본다.
- 모듈을 커널에서 제거한다.

【학습정리】

1. 디바이스 드라이버 모듈

* 문자 디바이스 드라이버의 기본 구성

- 문자 디바이스 드라이버의 제작 최소 항목
- 커널 소스 헤더파일
- 저소준 파일 입출력에 대응하는 file_operations 구조체에 등록할 함수
- file_operations 변수
- 문자 디바이스 드라이버를 등록하는 모듈 초기화 함수
- 문자 디바이스 드라이버를 제거하는 모듈 마무리 함수

2. 디바이스 드라이버파일 생성

- 디바이스 드라이버 구현 및 테스트 과정

