14주차 2차시 디바이스 드라이버 구성

【학습목표】

1. 문자 디바이스 드라이버의 기본 구성을 설명할 수 있다.
2. 디바이스 드라이버의 종류를 설명할 수 있다.

학습내용1 : 문자 디바이스 드라이버

1. file 구조체

- 디바이스 드라이버 구현시 사용
- c 라이브러리에 정의된 FILE과는 다름.
  - 응용프로그램에서는 사용할 수 없고, 단지 커널에서만 사용
- 파일 연산을 위한 인자 전달 방법으로 사용
- 디바이스에 필요한 자료 구조를 정의

```
struct file {
  structdentryf_dentry;                    //파일에 대한 디렉토리 엔트리
  structvfsmount *f_vfsmnt;
  structfile_operationsf_op;               // 다양한 파일 연산 함수가 정의된 파일 연산 구조체를 지시
  atomic_tf_count;                         // 프로세스 파일 참조 횟수
  unsigned intf_flags;                     // 접근모드, 비블록킹, 새성, 첨부 등에 관련된 플래그
  mode_t f_mode;                           // 파일 모드로 FMODE_READ or FMODE_WRITE비트에 의해 구
                                              분
  loff_t f_pos;                            // 현재 읽기/쓰기 위치
  unsigned long f_reada, f_ramax, f_raend, f_ralen, f_rawin;
  structfown_structf_owner;
  unsigned intf_uid, f_gid;        // 파일 사용자 및 그룹 식별자
  intf_error;
  unsigned long f_version;
  void *private_data;                      // 시스템 호추간에 필요한 정보를 보존하기 위해 사용
};
```

2. 파일 연산 구조체

- 응용프로그램은 저수준 파일 입출력 함수를 사용하여 디바이스 파일에 접근
- 커널은 등록된 디바이스 드라이버의 파일 연산 구조체를 참고해 응용프로그램의 요청에 대응하는 함수 호출
- 파일 연산 구조체 변수를 커널에 등록하는 것이 디바이스 드라이버 등록하는 것
- <linux> /include/linux/fs.h에 선언

```
structfile_operations {
 struct module *owner;                    //파일 작업의 소유자, 보통 THIS_MODULE로 지정
 loff_t (*lseek) (struct file *, loff_t, int);          //파일에서 현재 읽고 쓰는 위치 이동
 ssize_t (*read) (struct file *, char *, size_t, loff_t *);   //디바이스에서 파일 읽음
 ssize_t (*write) (struct file *, const char *, size_t, loff_t *);    //디바이스에 데이터 씀
 int (*readdir) (struct file *, void *, filldir_t);    //디렉토리에서만 사용 함수
 unsigned int (*poll) (struct file *, structpoll_table_struct *);       // 현 프로세스를 대기 큐에 넣음
 int (*ioctl) (structinode *, struct file *, unsigned int, unsigned long); //디바이스에 종속적인 명령을 만듬. 사용자가 임의로 함수 만듬
 int (*mmap) (struct file *, structvm_area_struct *);       // 디바이스의 메모리를 프로세스의 메모리로 매핑
 int (*open) (structinode *, struct file *);          // 디바이스를 개방
 int (*release) (structinode *, struct file *);       // 다비이스를 종료
 int (*fsync) (struct file *, structdentry *);        // 디바이스를 플러시함. 데이터 중에서 디바이스에 있는 것은 모두 디바이스에 씀
 int (*fasync) (int, struct file *, int);             // FASYNC 플래그 변화에 있는 디바이스 확인 용
 int (*check_media_change) (kdev_t dev);          // 미디어가 변경될 수 있는 블록비다이스 사용
 int (*revalidate) (kdev_t dev);              // 제거가능한 블록 디바이스에서 버퍼 캐시 관리용
 int (*lock) (struct file *, int, structfile_lock *);   // 파일에 잠근 기능
};
```

학습내용2 : 전형적인 문자 디바이스 드라이버

- **문자 디바이스 드라이버의 파일 연산 구조체 사용**
- **파일 연산 구조체에서 필요한 필드만 선언하면 된(선언하지 않은 필도는 NUL 처리)**
- **전형적인 파일 연산 구조체의 선언내용 (xxx는 일반적으로 디바이스의 이름)**

```
struct file_operations xxx_fops = {
        .owner = THIS_MODULE,
        .open = xxx_open,
        .release = xxx_release,
        .read = xxx_read,
        .write = xxx_write,
        .ioctl = xxx_ioctl,
}
```

1. 디바이스 드라이버의 등록과 해제

* 문자 디바이스의 등록과 해제

```
intregister_chrdev(unsigned int major, const char *name,  structfile_operations *fops)
intunregister_chrdev(unsigned int major,  const char *name)
```

- major : 주번호로, 0이면 사용하지 않는 주번호 중에서 동적으로 할당
- name : 디바이스의 이름으로 /proc/devices에 나타나 있음
- 성공 시 : 0이거나 양수
- 실패 시 : 음수

* 블록 디바이스의 등록과 해제

```
intregister_blkdev(unsigned int major, const char *name)
intunregister_blkdev(unsigned int major, const char *name)
```

- major : 주번호로, 0이면 사용하지 않는 주번호 중에서 동적으로 할당
- name : 블록 디바이스 장치 이름
- 성공 시 : 0이거나 양수
- 실패 시 : 음수

* 네트워크 디바이스의 등록과 해제

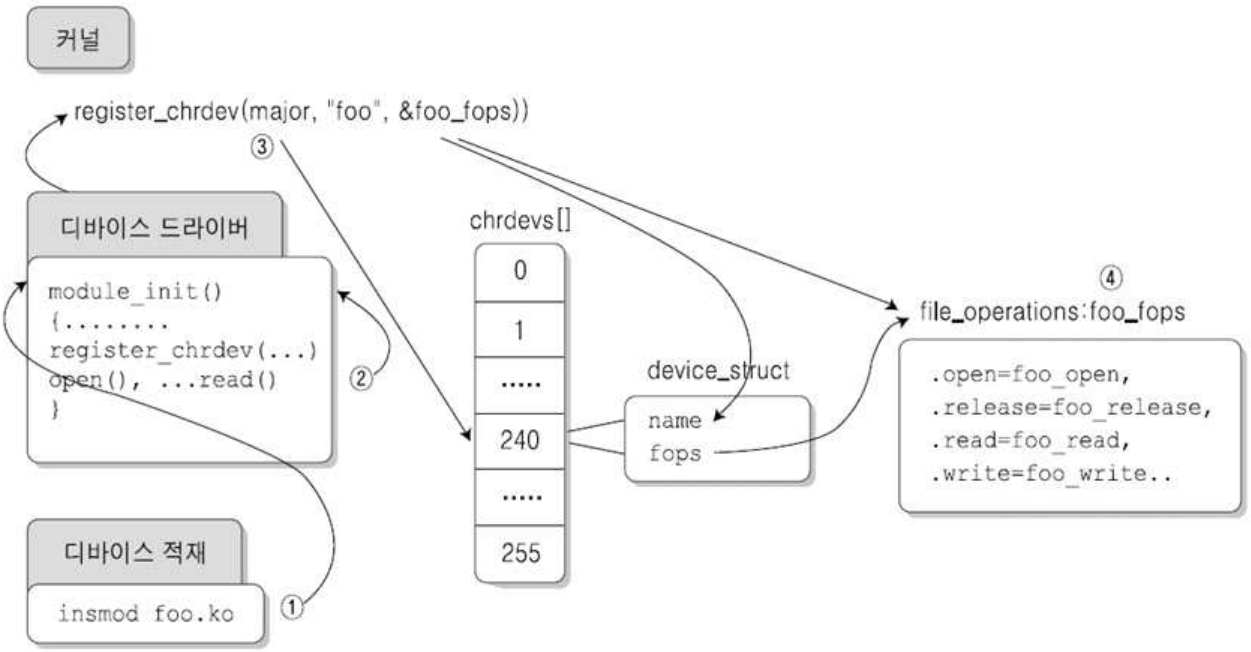
```
intregister_netdev(structnet_device *dev)
void unregister_netdev(structnet_device *dev)
```

- dev : net_device 구조체 이름
- 성공 시 0
- 실패 시 0이 아닌 값을 반환

* 문자 디바이스 드라이버의 등록 과정
- 커널 내부의 chrdevs[]배열 구조체에 적재될 디바이스 파일의 주번호가 할당되어야 한다.

```
struct char_device_struct chrdevs[MAX_PROBE_HASH];
```

- 쉘 모드에서 insmod foo.ko 실행
- insmod 명령어는 디바이스 드라이버 파일의 module_init() 호출
- resister_chrdev() 함수 실행, 주번호를 chrdevs[]배열의 인덱스로 사용해 배열에 접근(커널 내부의 chrdevs[]배열 구조체에 등록)
- foo_fops는 모듈내에 파일 연산 가리킴.디바이스 드라이버에 대응하는 함수와 연관



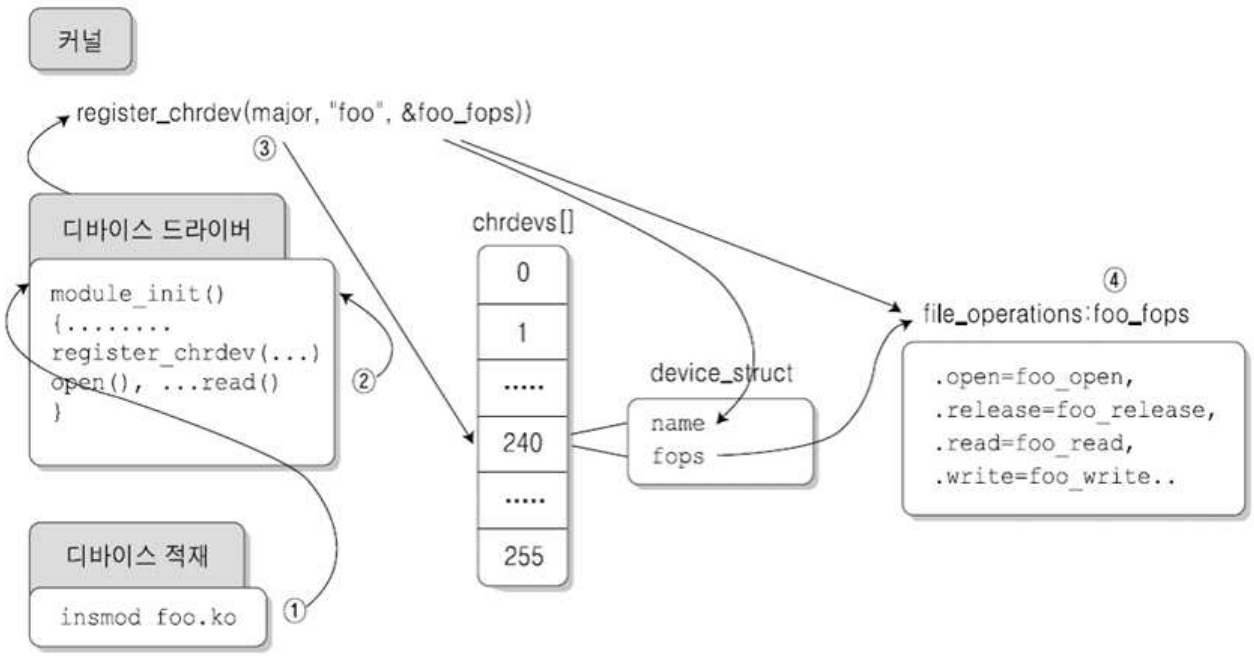
[그림 10-5] 디바이스의 등록과 파일 연산

【학습정리】

1. 문자 디바이스 드라이버
- 디바이스 드라이버 구현시 사용
- c 라이브러리에 정의된 FILE과는 다름. : 응용프로그램에서는 사용할 수 없고, 단지 커널에서만 사용
- 파일 연산을 위한 인자 전달 방법으로 사용
- 디바이스에 필요한 자료 구조를 정의

2. 전형적인 문자 디바이스 드라이버
- 문자 디바이스 드라이버의 파일 연산 구조체 사용
- 파일 연산 구조체에서 필요한 필드만 선언하면 된(선언하지 않은 필도는 NUL 처리)
- 전형적인 파일 연산 구조체의 선언내용 (xxx는 일반적으로 디바이스의 이름)

```
struct file_operations xxx_fops = {
        .owner = THIS_MODULE,
        .open = xxx_open,
        .release = xxx_release,
        .read = xxx_read,
        .write = xxx_write,
        .ioctl = xxx_ioctl,
}
```



[그림 10-5] 디바이스의 등록과 파일 연산