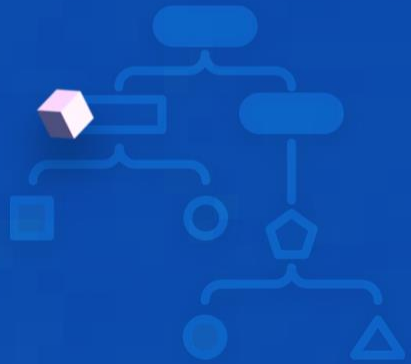
강원혁신플랫폼

리눅스프로그래밍

커널 모듈 프로그래밍



커널 모듈의 초기화 함수를 정의하는 데 사용되는 매크로입니다. <linux/init.h> 헤더 파일에 정의되어 있으며 모듈이 커널에 로드될 때 실행되어야 하는 함수를 지정하는 데 사용되는 매크로는 무엇인가요?

module_init()

학습목표

학습 내용

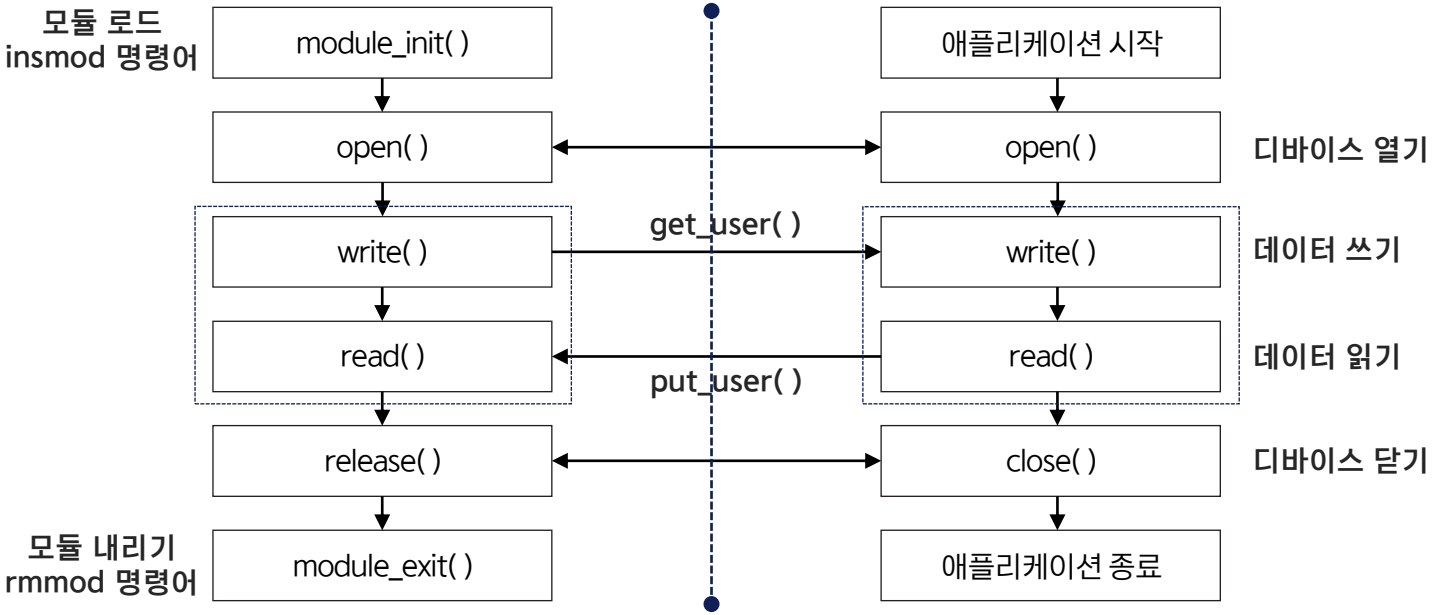
1. 커널 모듈의 동작
2. file_operations 구조체

학습 목표

- 커널 모듈의 동작을 설명할 수 있다.
- file_operations 구조체에 대해 설명할 수 있다.

커널 모듈의 동작

커널 모듈의 동작


모듈 로드
insmod 명령어
module_init( )
open( )
write( )
read( )
release( )
모듈 내리기
rmmod 명령어
module_exit( )
애플리케이션 시작
open( )
write( )
read( )
close( )
애플리케이션 종료
get_user( )
put_user( )
디바이스 열기
데이터 쓰기
데이터 읽기
디바이스 닫기

리눅스 디바이스 드라이버 관리 주요 명령어

01 02 03 04 05 06

insmod

- 커널 모듈을 로드하는 데 사용
- 디바이스 드라이버를 컴파일하여 .ko 파일을 생성한 후, insmod 명령어를 사용하여 커널에 모듈을 로드
- insmod mydriver.ko

리눅스 디바이스 드라이버 관리 주요 명령어

rmmod

- 커널 모듈을 제거하는 데 사용
- 로드된 디바이스 드라이버를 제거하기 위해 rmmod 명령어를 사용
- rmmod mydriver

리눅스 디바이스 드라이버 관리 주요 명령어

modinfo

- 커널 모듈에 대한 정보를 표시
- 모듈의 이름, 저작권자, 버전, 설명 등을 확인
- modinfo mydriver

리눅스 디바이스 드라이버 관리 주요 명령어

01 02 03 04 05 06

lsmod

- 현재 로드된 커널 모듈을 나열
- 디바이스 드라이버가 정상적으로 로드되었는지 확인
- lsmod | grep mydriver

리눅스 디바이스 드라이버 관리 주요 명령어

01 02 03 04 05 06

dmesg

- 커널 메시지 버퍼를 표시
- 로드된 디바이스 드라이버와 관련된 로그 메시지를 확인
- dmesg | grep mydriver와 같이 사용하여 특정 드라이버와 관련된 로그를 검색

리눅스 디바이스 드라이버 관리 주요 명령어

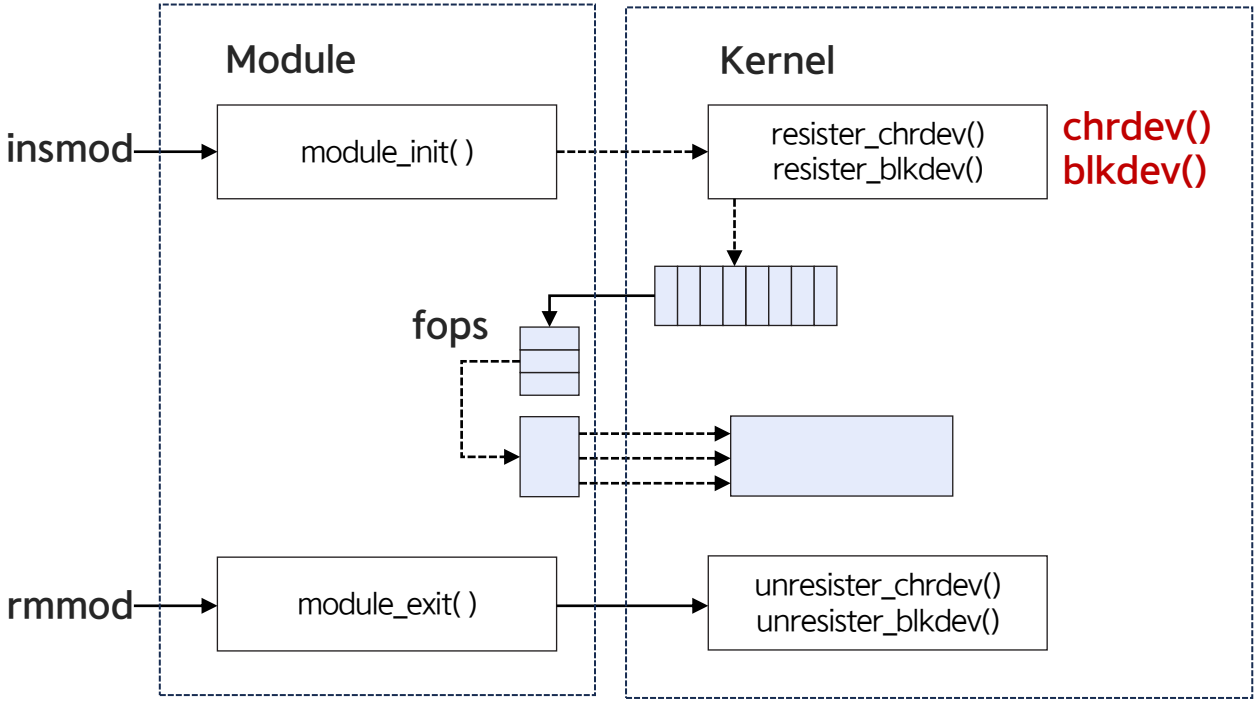
01 02 03 04 05 06

mknod

- 디바이스 파일을 생성
- 디바이스 드라이버가 새로운 디바이스 파일을 필요로 할 때 사용
- mknod /dev/mydevice c 250 0

file_operations 구조체

insmod / rmmod



출처 https://cms3.koreatech.ac.kr/sites/joo/IFC414/IFC414_10.pdf

커널 모듈 기본 구성

기본 헤더 (mydev 디바이스)

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
....
int mydev_open(…) {…} // Functions
int mydev_release(…) {…}
ssize_t mydev_write(…) {…}
ssize_t mydev_read(…) {…}

static struct file_operations mydev_fops = { // File Operations
.open = mydev_open,
```

커널 모듈 기본 구성

기본 헤더 (mydev 디바이스)

```
.release = mydev_release,
.write = mydev_write,
.read = mydev_read,
.llseek = NULL
};

module_init(mydev_init); // insmod
module_exit(mydev_exit); // rmmod
```

file_operations 구조체

file_operations 구조체
./include/linux/fs.h 참조

```
static struct file_operations mydev_fops = {
.owner = THIS_MODULE,
.open = mydev_open,
.release = mydev_release,
.write = mydev_write,
.read = mydev_read,
.unlocked_ioctl = mydev_ioctl,
.llseek = NULL
};
```

커널 모듈 hello_module.c


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

MODULE_LICENSE("Dual BSD/GPL");

/* 모듈의 초기화 부분 */
static int initModule(void)
{
    printk(KERN_INFO "Hello module!\n");
    return 0;
}
```

커널 모듈 hello_module.c


```
/* 모듈이 내려질 때의 정리 부분 */
static void cleanupModule(void)
{
    printk(KERN_INFO "Good-bye module!\n");
}

/* 모듈 함수 등록 부분 */
module_init(initModule);
module_exit(cleanupModule);
```

리눅스 커널 모듈, Makefile


```
KDIR = /lib/modules/`uname -r`/build
obj-m := hello_module.o

default:
	$(MAKE) -C $(KDIR) M=$$PWD modules

clean:
	$(MAKE) -C $(KDIR) M=$$PWD clean
```

```
$ make
$ ls
$ modeinfo hello_module.ko
```

리눅스 커널 모듈 사용

- 모듈 커널에 추가

```
$ sudo insmod ./hello_module.ko
```

- 모듈 확인

```
$ lsmod
```

- 모듈 제거

```
$ sudo rmmod hello_module
```

- 모듈 메시지 확인

```
$ dmesg | tail
[162255.769441] Hello module!
[162334.184292] Good-bye module!
```

요약정리

01 • 커널 모듈의 동작

02 • file_operations 구조체