KWANGWOON UNIVERSITY

2022년 2학기 **운영체제실습** 5주차

Module Programming, Wrapping

System Software Laboratory
School of Computer and Information Engineering
Kwangwoon Univ.

Contents

- **Module Programming**
  - 모듈의 이해
  - 특징
  - 모듈 프로그래밍 절차
  - 커널 모듈 구성
  - 커널 모듈의 추가 및 제거

- **실습**
  - 모듈 Load / Unload
  - Wrapping을 통한 Module Programming

KWANGWOON UNIVERSITY

Module Programming

System Software Laboratory
School of Computer and Information Engineering
Kwangwoon Univ.

Kernel Module

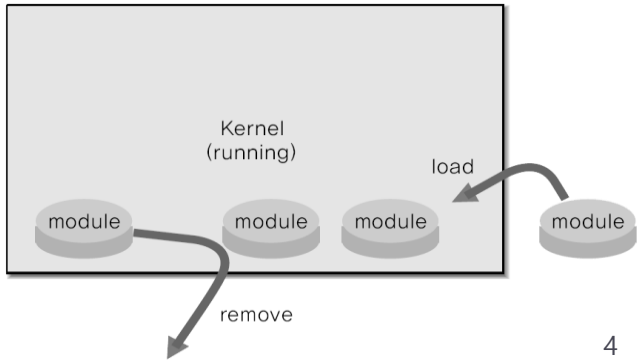
- **Kernel Module**
  - 커널 코드의 일부를 커널이 동작하는 상태에서 로드 또는 언 로드 가능
  - 커널 크기 최소화, 유연성 제공
    - 커널이 실행 중에 동적으로 로딩하여 커널과 링크함으로써 커널의 기능을 확장하여 사용할 수 있다.
    - 불필요 시에 커널과의 링크를 풀고 메모리에서 제거 할 수 있다.

      ➔ 커널 재 컴파일 없이 커널 기능 확장 가능

  - 각종 디바이스 드라이버를 사용할 때 유용
    - 마우스, 키보드, 사운드카드 드라이버는 종류가 다양하고 상황에 따라 사용하지 않을 수 있기 때문

      ➔ 새로운 장치를 추가할 때마다 커널을 재 컴파일 한다면?

  - 파일시스템, 통신 프로토콜 및 시스템 콜 등도 모듈로 구현 가능

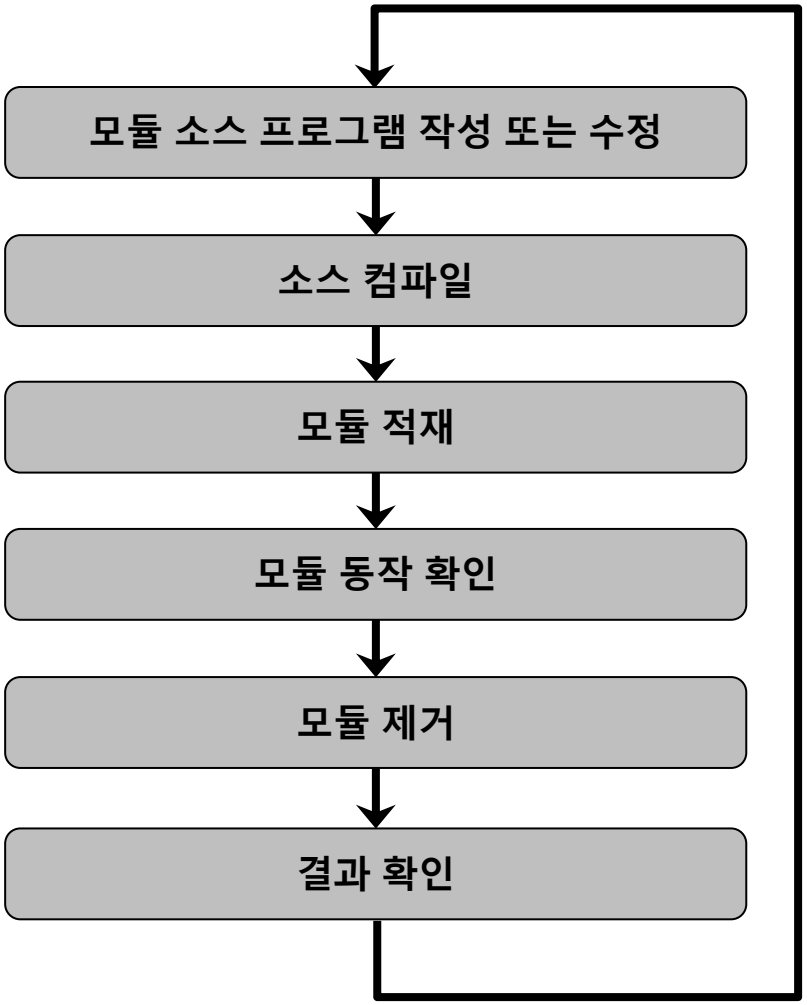


Kernel Module

- **특징**
  - 사건 구동형(event-driven program) 방식으로 작성
  - 명시적인 커널 모듈 설치 및 제거 과정이 필요
    - insmod, rmmod 명령어 등
  - 내부 main() 함수 없음
  - 디바이스 드라이버, 파일시스템, 네트워크 프로토콜 스택 등에 적용
    - 커널 경량화를 위해 반드시 필요
    - 임베디드 시스템의 경우, 제한적인 자원으로 인해 커널 등 시스템 소프트웨어의 최소화가 필요
  - 외부로 공개할 전역변수 사용에 주의
  - 커널에 적재 모듈 프로그램은 무제한의 특권을 가지므로 신중하게 작성해야 함.

Kernel Module

- 모듈 프로그래밍 절차



커널 모듈의 추가 및 제거

- **커널 모듈 추가**
  - 커널 모듈이 적재되면 오브젝트 파일의 내용이 커널 영역으로 복사
  - module_init() 에 명시한 함수를 호출하여 커널 모듈 등록
  - 설치된 모듈은 '/proc/modules' 파일에 기록
    - $ cat /proc/modules 로 확인 가능

- **커널 모듈을 제거**
  - 커널 모듈이 제거되면 module_exit() 에 명시한 함수를 호출
  - module_init() 에서 호출한 함수에서 할당한 자원을 반환
  - 커널 모듈의 등록 해제
  - 커널 모듈의 오브젝트 코드를 위해 할당했던 메모리를 반환

커널 모듈의 추가 및 제거

- **커널 모듈 추가 및 제거 예시**
  - module_init, module_exit 매크로 지원
    - module_init() : startup 함수 (모듈을 로드하면 해당 매크로에 명시된 함수를 호출)
    - module_exit() : cleanup 함수 등록 (언 로드시 등록된 함수 호출)

```
#include <linux/module.h>
/* global variables */
...
static int __init module_start() {
/* 모듈이 설치될 때에 초기화를 수행하는 코드 */ }
static int __exit module_end() {
/* 모듈이 제거될 때에 반환작업을 수행하는 코드 */ }

module_init(module_start);
module_exit(module_end);
...
```

insmod or modprobe

rmmod

커널 모듈의 추가 및 제거

- **커널 모듈 구성 (make)**
  - 모듈 프로그램의 Makefile
    - 모듈 생성을 위한 일반적인 Makefile

```
obj-m := test.o #module object name

KDIR := /lib/modules/$(shell uname -r)/build #kernel module directory
PWD := $(shell pwd) #cwd

all:
        $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules #-C is change directory opt.

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

- obj-m := test.o ➔ 모듈로 생성할 이름 정의 (test)
- KDIR ➔ 커널 코드 디렉토리 위치 (symbolic link)
- $(shell uname –r) ➔ 현재 **실행** 중인 커널 버전
- PWD ➔ 컴파일 대상이 되는 모듈 코드가 있는 위치(test.c 위치)
- default ➔ (target) 모듈을 컴파일 하는 명령
- clean ➔ (target) 컴파일 결과로 생성된 파일 모두 지움

커널 모듈의 추가 및 제거

- **사용 명령어**

이름	용도
insmod	simple program to insert a module into the Linux Kernel (load)
rmmod	simple program to remove a module from the Linux Kernel (unload)
lsmod	program to show the status of modules in the Linux Kernel
depmod	program to generate modules.dep and map files (디스크 내 적재된 커널 모듈 간 의존성 검사)
modprobe	program to add and remove modules from the Linux Kernel (insmod와 유사하나, 모듈간 의존성을 검사하여 그 결과 누락된 다른 모듈을 찾아서 적재)
modinfo	program to show information about a Linux Kernel module

예제 – Module Load / Unload

```c
#include <linux/module.h>

static int __init test_init(void)
{
        printk(KERN_INFO "insmod! %lld\n", get_jiffies_64());

        return 0;
}

static void  __exit test_exit(void)
{
        printk(KERN_INFO "rmmod! %lld\n", get_jiffies_64());
}

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

test.c

```
obj-m := test.o    #module object name

KDIR := /lib/modules/$(shell uname -r)/build    #kernel module directory
PWD := $(shell pwd) #cwd

all:
        $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules #C is change directory opt
clean:
        $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```

Makefile

예제 – Module Load / Unload

- 컴파일

```
sslab@ubuntu:~/moduleTest$ sudo make
make -C /lib/modules/4.19.67-SSLAB/build SUBDIRS=/home/sslab/moduleTest  modules

make[1]: Entering directory '/home/sslab/Downloads/linux-4.19.67'
  CC [M]  /home/sslab/moduleTest/test.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/sslab/moduleTest/test.mod.o
  LD [M]  /home/sslab/moduleTest/test.ko
make[1]: Leaving directory '/home/sslab/Downloads/linux-4.19.67'
```

- 모듈 적재, 확인 및 제거

```
sslab@ubuntu:~/moduleTest$ ls
Makefile       Module.symvers  test.ko     test.mod.o
modules.order  test.c          test.mod.c  test.o
sslab@ubuntu:~/moduleTest$ sudo insmod test.ko
sslab@ubuntu:~/moduleTest$ lsmod |grep test
test                   16384  0
sslab@ubuntu:~/moduleTest$ sudo rmmod test
sslab@ubuntu:~/moduleTest$ sudo dmesg |tail -n -2
[ 1809.399421] insmod! 4295344597
[ 1825.503046] rmmod! 4295348624
```

KWANGWOON UNIVERSITY

Wrapping

System Software Laboratory
School of Computer and Information Engineering
Kwangwoon Univ.

System Call wrapping을 위한 Module Programming

- **Hooking**
  - 운영체제, 응용 프로그램 등에서 동작하는 함수 등을 다른 것으로 대체하는 행위
  - 본 수업에서는 Linux 커널 내 존재하는 기존의 시스템 콜을 다른 행동을 하는 시스템 콜로 hooking
    - 시스템 콜 hooking 작업을 kernel module에서 수행

System Call wrapping을 위한 Module Programming

- **Example**
  - Systemcall add hooking
    - 4주차 강의자료 참고

```
546     x32     preadv2                 __x32_compat_sys_preadv64v2
547     x32     pwritev2                __x32_compat_sys_pwritev64v2
548     common  hello                   __x64_sys_hello
549     common  add                     __x64_sys_add
```

```
sslab@ubuntu:~$ mkdir hookingTest
sslab@ubuntu:~$ cd hookingTest/
sslab@ubuntu:~/hookingTest$ vi hooking.c
```

System Call wrapping을 위한 Module Programming

- **Example**
  - Systemcall add hooking
    - hooking.c 구조

```
#include <linux/module.h>
#include <linux/highmem.h>
#include <linux/kallsyms.h>
#include <linux/syscalls.h>
#include <asm/syscall_wrapper.h>

#define __NR_add 549

void **syscall_table;
void *real_add;

__SYSCALL_DEFINEx(2, sub, int, a, int, b){
                .....
}
void make_rw(void *addr) {
                .....
}
void make_ro(void *addr){
                .....
}
static int __init hooking_init(void){
                .....
}
static void __exit hooking_exit(void){
                .....
}
module_init(hooking_init);
module_exit(hooking_exit);
MODULE_LICENSE("GPL");
```

기존 시스템콜을 대체할 시스템 콜 정의

addr이 속해 있는 페이지의 읽기 및 쓰기 권한을 부여하는 함수

addr이 속해 있는 페이지의 읽기 및 쓰기 권한을 회수하는 함수

모듈 적제 시 호출되는 함수

모듈 해제 시 호출되는 함수

System Call wrapping을 위한 Module Programming

```
1  #include <linux/module.h>
2  #include <linux/highmem.h>
3  #include <linux/kallsyms.h> /* kallstms_lookup_name() */
4  #include <linux/syscalls.h> /* __SYSCALL_DFINRx() */
5  #include <asm/syscall_wrapper.h> /*__SYSCALL_DFINRx() */
6
7
8  #define __NR_add 549
9
10 void **syscall_table;
11 void *real_add;
12
13
14
15 __SYSCALL_DEFINEx(2, sub, int, a, int, b)
16 {
17         printk("kernel hooked func : %d - %d\n", a, b);
18
19         return a-b;
20 }
```

- **Parametres of this macro**
  - 2 ➔ 시스템 콜의 파리미터 수
  - sub ➔ 새로운 시스템 콜
  - int ➔ 첫 번째 파라미터의 유형
  - a ➔ 첫 번째 파라미터
  - int ➔ 두 번째 파라미터 유형
  - b ➔ 두 번째 파라미터

- **Example**
  - hooking.c
    - **10: void **syscall_table;**
      - 기존의 add 시스템 콜 주소를 저장 할 포인터
    - **15: __SYSCALL_DEFINEx(2, sub, int, a, int, b)**
      - 기존의 add 시스템 콜을 대체할 sub 시스템 콜 정의
      - SYSCALL_DEFINE*n*() 매크로 (단, $0 \le n \le 6$) 에서 사용
      - 단순히 함수 정의 뿐만 아니라 시스템 콜 호출을 위한 관련 작업을 일괄 수행해주는 매크로로, 이를 통해 정의해야 후킹 가능

System Call wrapping을 위한 Module Programming

```
22 void make_rw(void *addr)
23 {
24         unsigned int level;
25         pte_t *pte = lookup_address((u64)addr, &level);
26         if(pte->pte &~ _PAGE_RW)
27                 pte->pte |= _PAGE_RW;
28
29 }
30
31 void make_ro(void *addr)
32 {
33         unsigned int level;
34         pte_t *pte = lookup_address((u64)addr, &level);
35
36         pte->pte = pte->pte &~ _PAGE_RW;
37 }
38
```

- **22: void make_rw(void *addr)**
  - addr이 속해 있는 페이지의 읽기 및 쓰기 권한을 부여하는 함수
  - 기본적으로, system call table은 쓰기 권한이 존재하지 않음
    - 본 함수를 호출하여 쓰기 권한이 없는system call table에 쓰기 권한을 부여

- **31: void make_ro(void *addr)**
  - addr이 속해 있는 페이지의 읽기 및 쓰기 권한을 회수

System Call wrapping을 위한 Module Programming

```
40 static int __init hooking_init(void)
41 {
42         /* Find system call table */
43         syscall_table = (void**) kallsyms_lookup_name("sys_call_table");
44
45
46         /*
47          * Change permission of the page of system call table
48          * to both readable and writable
49          */
50         make_rw(syscall_table);
51         real_add = syscall_table[__NR_add];
52         syscall_table[__NR_add] = __x64_syssub;
53
54         return 0;
55 }
```

- **40: static int __init hooking_init(void)**
  - 모듈 적재 시 호출되는 함수
- **43: syscall_table =(void**) kallsyms_lookup_name("sys_call_table");**
  - system call table의 주소를 찾는 함수
    - "sys_call_table" 이라는 전역 변수로 선언되어 있는 시스템 콜 위치를 찾음
- **50: make_rw(syscall_table);**
  - 쓰기 금지되어 있는 시스템 콜 테이블에 쓰기 권한 부여
- **51: real_add = syscall_table[__NR_add];**
  - 모듈 해제 시 기존의 시스템 콜을 원상 복구 하기 위해, 기존 시스템 콜 주소 저장
- **52: syscall_table[__NR_add]= __x64_syssub;**
  - 후킹할 "sub" 시스템 콜을 add 시스템 콜 대신 대체하는과정
  - Line 15의 결과로 __x64_syssub함수가 생성되며, 이를 삽입

System Call wrapping을 위한 Module Programming

```
58 static void __exit hooking_exit(void)
59 {
60         syscall_table[__NR_add] = real_add;
61         /* Recover the page's permission (i.e. read-only)*/
62         make_ro(syscall_table);
63
64 }
65
66 module_init(hooking_init);
67 module_exit(hooking_exit);
68 MODULE_LICENSE("GPL");
69
```

- **58: static void __exit hooking_exit(void)**
  - 모듈 해제 시 호출되는 함수
- **60: syscall_table[__NR_add] = real_add;**
  - 후킹했던 시스템 콜을 원래대로 복원하는 작업

System Call wrapping을 위한 Module Programming

- **Example**
  - Makefile (for module)

```
obj-m := hooking.o

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

all:
	$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
	gcc -o app app.c
clean:
	$(MAKE) -C $(KDIR) SUBDIRS=$(PWD) clean
```

  - app.c (test application)

```
 1 #include <stdio.h>
 2
 3 #include <unistd.h> /* syscall() */
 4 #include <sys/syscall.h> /* syscall() */
 5
 6 int main(void)
 7 {
 8     int a, b;
 9     long ret;
10
11     a = 7;
12     b = 4;
13
14     ret = syscall(549, a, b);
15     printf("%d op. %d = %ld\n", a, b, ret);
16
17     return 0;
18 }
```

System Call wrapping을 위한 Module Programming

- **Example**
  - Results

```
sslab@ubuntu:~/hookingTest$ sudo make
make -C /lib/modules/4.19.67-SSLAB/build SUBDIRS=/home/sslab/hookingTest modules
make[1]: Entering directory '/home/sslab/Downloads/linux-4.19.67'
  CC [M]  /home/sslab/hookingTest/hooking.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/sslab/hookingTest/hooking.mod.o
  LD [M]  /home/sslab/hookingTest/hooking.ko
make[1]: Leaving directory '/home/sslab/Downloads/linux-4.19.67'
gcc -o app app.c
sslab@ubuntu:~/hookingTest$ ls
app     hooking.c   hooking.mod.c  hooking.o  modules.order
app.c   hooking.ko  hooking.mod.o  Makefile   Module.symvers
sslab@ubuntu:~/hookingTest$
```

```
sslab@ubuntu:~/hookingTest$ ls
app     hooking.c   hooking.mod.c  hooking.o  modules.order
app.c   hooking.ko  hooking.mod.o  Makefile   Module.symvers
sslab@ubuntu:~/hookingTest$ ./app    ①
7 op. 4 = 11
sslab@ubuntu:~/hookingTest$ sudo insmod hooking.ko
sslab@ubuntu:~/hookingTest$ ./app
                                     ②
7 op. 4 = 3
sslab@ubuntu:~/hookingTest$ lsmod |grep hooking
hooking                16384  0
sslab@ubuntu:~/hookingTest$ sudo rmmod hooking.ko
sslab@ubuntu:~/hookingTest$ ./app    ③
7 op. 4 = 11
sslab@ubuntu:~/hookingTest$
```

Assignment 2

- 제출 기한: 2022. 09.22(목) ~ 2022.10.13(목) 23:59:59
- Delay 없음
- 업로드 양식에 어긋날 경우 감점 처리
- Hardcopy 제출하지 않음