KWANGWOON UNIVERSITY

2024년 2학기 **운영체제실습**

Assignment 2

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

Contents

- **Assignment2-1**
  - Make os_ftrace system call
- **Assignment2-2**
  - Make Wrapping & hooking modules
- **Assignment2-3**
  - Make trace modules

Assignment 2

- **Step by step**
  - Assignment 2-1
    - ftrace 시스템 콜 “os_ftrace”을 만든다.
  - Assignment 2-2
    - os_ftrace 시스템 콜을 hijack 하여 my_ftrace 함수로 대체한다.
  - Assignment 2-3
    - os_ftrace 시스템 콜을 hijack 하여 my_ftrace 함수로 대체한다.
    - openat, read, write, lseek, close 시스템 콜의 원형을 찾는다.
    - openat / read / write / lseek / close 시스템콜을 hijack하여 ftrace_openat / ftrace_read / ftrace_write / ftrace_lseek / ftrace_close 함수로 대체한다.

Assignment 2

- **Warning**
  - Grub 에서 nokaslr 설정 되어 있는지 확인

```
KASLR disabled: 'nokaslr' on cmdline.
[    2.184933] sd 2:0:0:0: [sda] Assuming drive cache: write through
/dev/sda5: clean, 348718/3899392 files, 11599990/15596800 blocks
[    3.262527] piix4_smbus 0000:00:07.3: SMBus Host Controller not enabled!
```

Booting시 확인 가능

  - $ vi etc/default/grub(**2주차 강의 자료 참조**)

```
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet nokaslr"
```

- **KASLR(Kernel Address Space Layout Randomization)**
  - 커널의 기본 주소 값을 무작위로 만듦.
  - $ cat /proc/kallsyms | grep sys_call_table (초반 주소 : 000… 으로 되어있음)

```
root@ubuntu:/home/os2024123456/hooking# cat /proc/kallsyms | grep sys_call_table
ffffffff82200240 R sys_call_table
ffffffff822015a0 R ia32_sys_call_table
```

  - $ cat /boot/System.map-$(uname -r) | grep sys_call_table

```
root@ubuntu:/home/os2024123456/hooking# cat /proc/kallsyms | grep sys_call_table
ffffffff82200240 R sys_call_table
ffffffff822015a0 R ia32_sys_call_table
```

  - **해당 두 주소가 동일해야 함.**

Assignment 2-1

- **Requirements**
  - ftrace 시스템콜 생성
    - `Asmlinkage int os_ftrace(pid_t pid);  //system call table number : 336번`
    - 커널 소스 폴더에 os_ftrace 폴더를 만들고 진행
  - os_ftrace 시스템 콜 출력양식
    - ORIGINAL ftrace( ) called! PID is [*#pid*]

```
os2024123456@ubuntu:~$ dmesg | tail -n 3
[  158.083789] ORIGINAL ftrace() called! PID is [2088]
[  194.272712] ORIGINAL ftrace() called! PID is [2130]
[  202.539611] ORIGINAL_ftrace() called! PID is [2132]
```

  - **Tip**
    - **3주차 운영체제 실습 강의자료 참조**

Assignment 2-1

- **Requirements**
  - 제출 파일
    - **1) os_ftrace.c**
      - 시스템 콜 코드 파일
  - 보고서
    - 해당 시스템 콜 구현 및 적용 **과정 및 결과화면 첨부**

Assignment 2-2

- **Requirements**
  - os_ftrace 시스템 콜 wrapping 하는 모듈 제작
    - Assignment 2-1 에서 만든 os_ftrace를 wrapping 하는 모듈 제작
    - [출력 예시]
      - 기존 os_ftrace 출력

```
os2024123456@ubuntu:~$ dmesg | tail -n 3
[  158.083789] ORIGINAL ftrace() called! PID is [2088]
[  194.272712] ORIGINAL ftrace() called! PID is [2130]
[  202.539611] ORIGINAL ftrace() called! PID is [2132]
```

      - 시스템 콜 wrapping 후 os_ftrace 출력

```
os2024123456@ubuntu:~$ dmesg | tail -n 3
[   213.490298] os_ftrace() hooked! os_ftrace -> my_ftrace
[   217.130272] os_ftrace() hooked! os_ftrace -> my_ftrace
[   220.445129] os_ftrace() hooked! os_ftrace -> my_ftrace
```

    - **Tip**
      - **4주차 운영체제 실습 강의자료 참조**

Assignment 2-2

- **Requirements**
  - 제출 파일
    - **1) os_ftracehooking.c**
      - `os_ftrace` 시스템콜을 hijack하여 `my_ftrace` 함수로 대체하는 커널 모듈 파일
  - 보고서
    - 해당 모듈 구현 및 적용 **과정 첨부**

Assignment 2-3

- **`ftrace` : File Tracing**
  - 특정 pid에 대하여 **파일에 관한 시스템콜을 추적**하는 툴 작성

  - 결과 예시

```
os2024123456@ubuntu:~/hooking$ dmesg | tail -n 4
[49724.719177] OS Assignment2 ftrace [6231] Start
[49724.719320] [2024123456] a.out file[abc.txt] stats [x] read - 5 / written - 26
[49724.719321] open[1] close[1] read[4] write[5] lseek[9]
[49724.719322] OS Assignment2 ftrace [6231] End
```

  - 추적할 시스템콜
    - `openat / read / write / lseek / close`

Assignment 2-3

- **Requirements**
  - Trace 시작
    - 출력없음 : Trace 시작 시점은 `ftrace` 시스템콜에 특정 pid 값을 넣고 호출
  - Trace 종료
    - 커널 log 출력
    - [출력 양식]

OS Assignment 2 ftrace [**pid**] Start
[학번] **process name** file[**file name**] stats [x] read – **read bytes** / written – **write bytes**
open[**open count**] close[**close count**] read[**read count**] write[**write count**] lseek[**lseek count**]
OS Assignmnet 2 ftrace [**pid**] End

```
os2024123456@ubuntu:~/hooking$ dmesg | tail -n 4
[49724.719177] OS Assignment2 ftrace [6231] Start
[49724.719320] [2024123456] a.out file[abc.txt] stats [x] read - 5 / written - 26
[49724.719321] open[1] close[1] read[4] write[5] lseek[9]
[49724.719322] OS Assignment2 ftrace [6231] End
```

    - trace 종료 시점 : `ftrace` 시스템콜에 0 값을 넣고 호출

Assignment 2-3

- **System call address**
  - $ cat /boot/System.map-$(uname –r) | grep “system_call_name”

```
os2024123456@ubuntu:~$ cat /boot/System.map-$(uname -r) | grep "__x64_sys_openat"
ffffffff813865f0 T __x64_sys_openat2
ffffffff81387d90 T __x64_sys_openat
```

- **About Open( )**
  - $ man open 2

```
  C library/kernel differences
       Since  version  2.26,  the  glibc  wrapper function for open() employs the openat() system call, rather
than the kernel's open() system call.  For certain architectures, this is also true in
       glibc versions before 2.26.
```

- **Check glibc version**
  - $ getconf -a | grep glibc

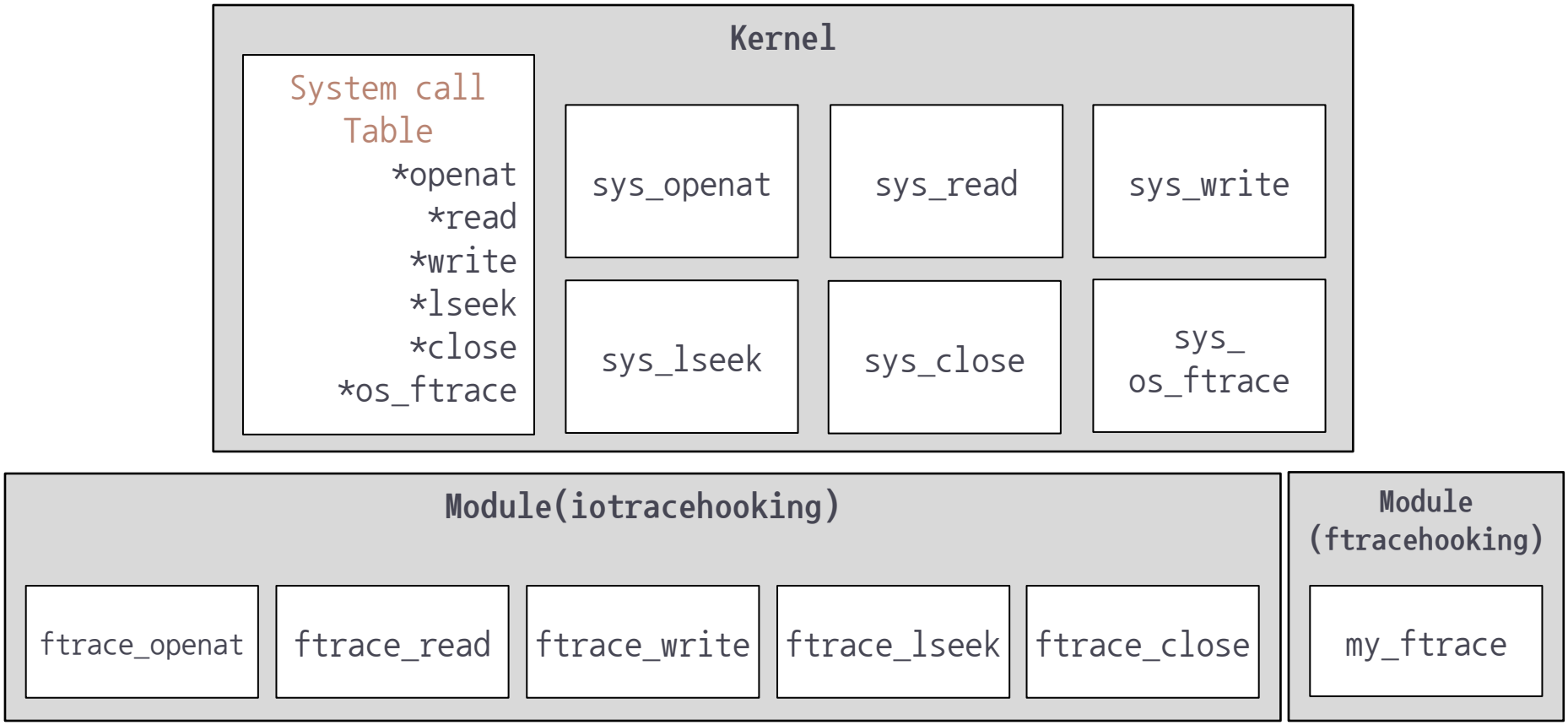
```
os2024123456@ubuntu:~$ getconf -a | grep glibc
GNU_LIBC_VERSION                   glibc 2.31
os2024123456@ubuntu:~$
```

Assignment 2-3

- **Step by step**
  - os_ftrace 시스템콜을 hijack 하여 my_ftrace 함수로 대체한다.
    - ftracehooking.c
      - static asmlinkage int my_ftrace(const struct pt_regs *regs) 사용
      - **iotracehooking.c와 연동**
        - EXPORT_SYMBOL() 사용
  - openat, read, write, lseek, close 시스템콜의 원형을 찾는다.
    - Cscope , ctags 이용
  - openat / read / write / lseek / close 시스템콜을 hijack하여 ftrace_openat / ftrace_read / ftrace_write / ftrace_lseek / ftrace_close 함수로 대체한다.
    - iotracehooking.c
    - ex. static asmlinkage long ftrace_openat(const struct pt_regs *regs) 사용

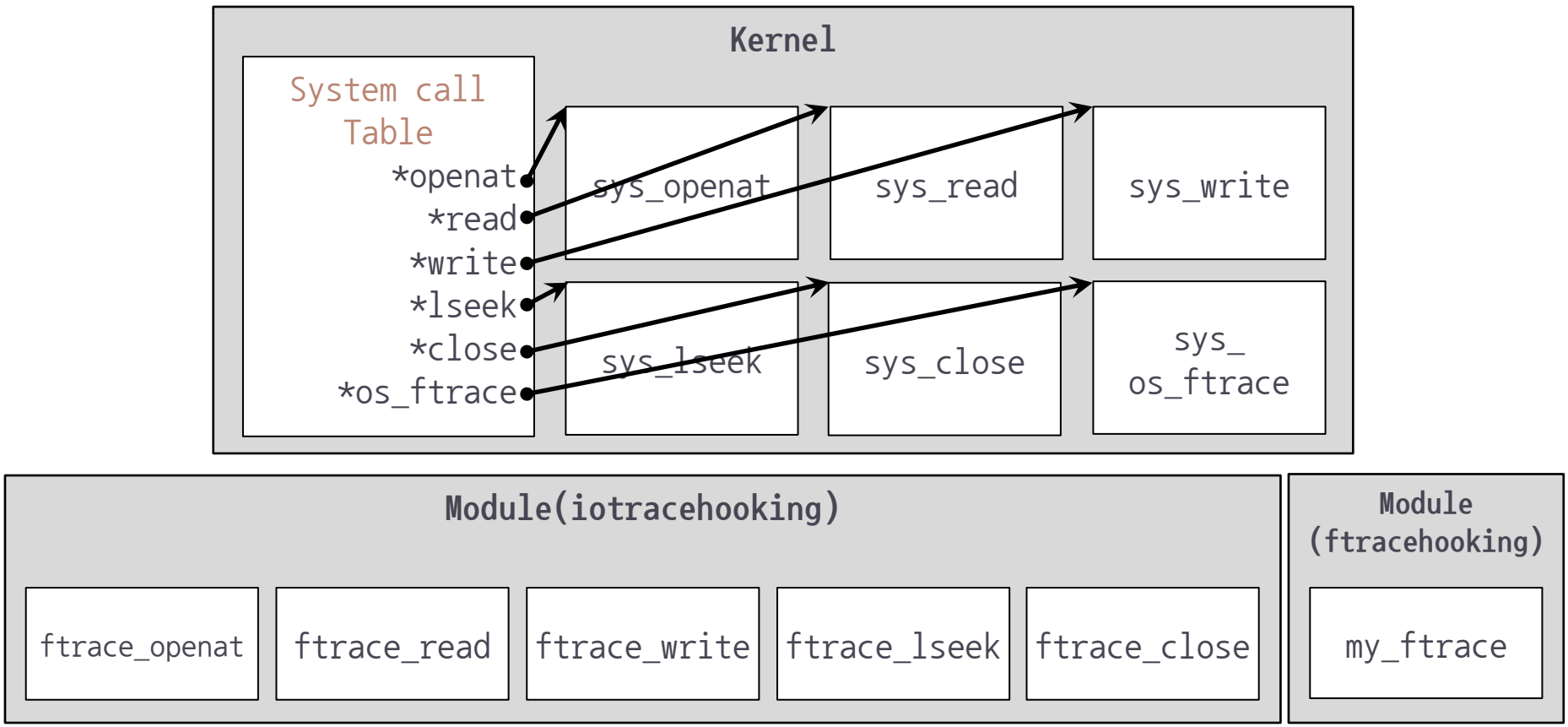
Assignment 2-3

- **Behaviors**
  - **Before** inserting modules



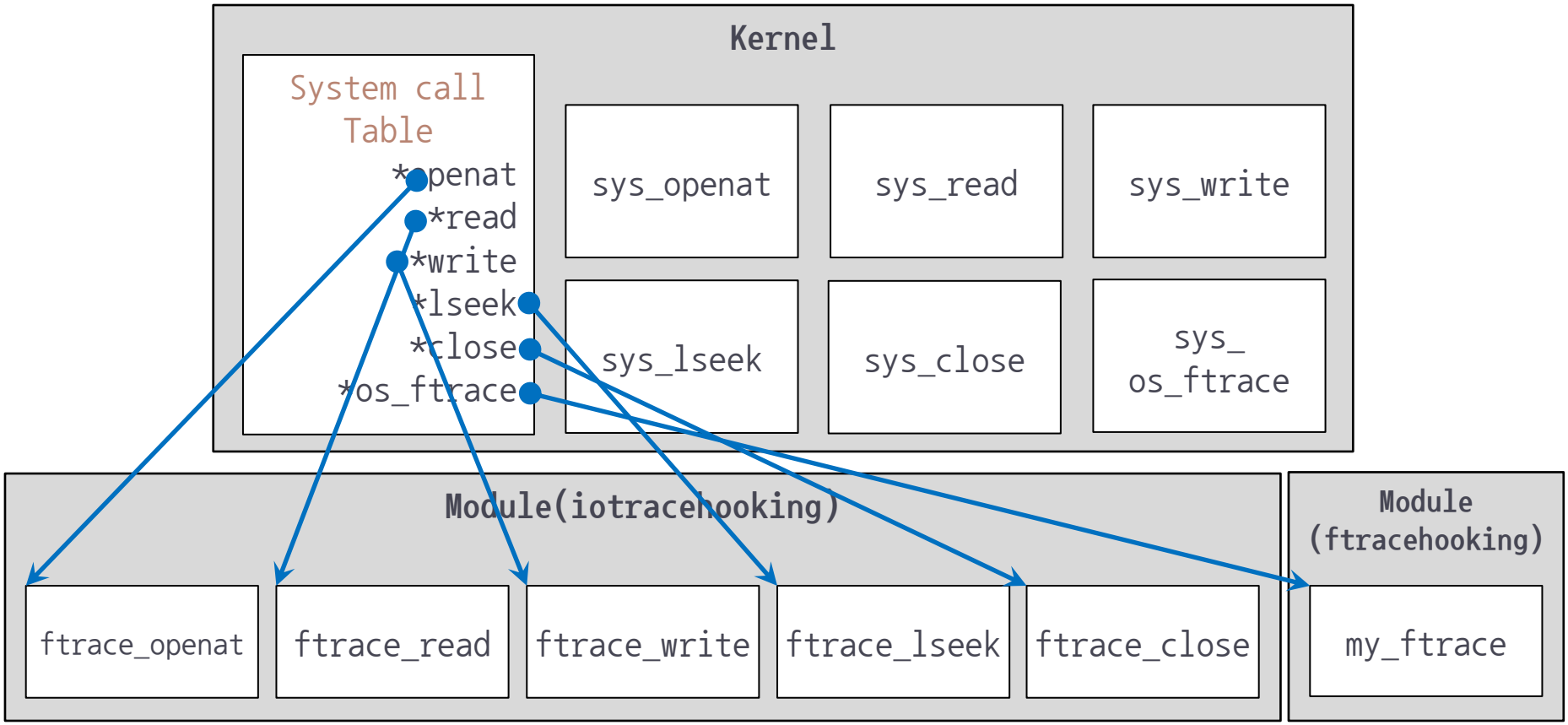
Assignment 2-3

- **Behaviors**
  - **Before** inserting modules



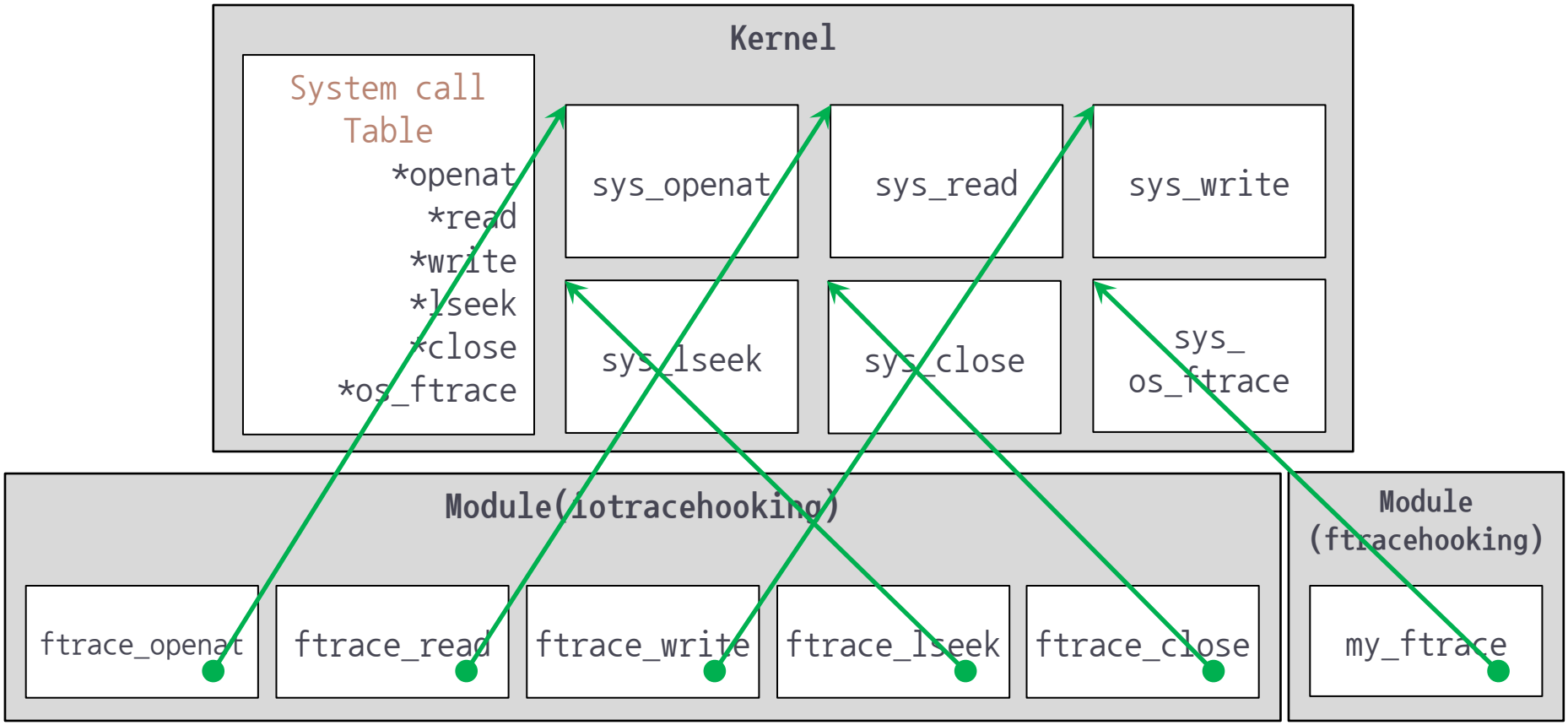
Assignment 2-3

- **Behaviors**
  - **After** inserting modules



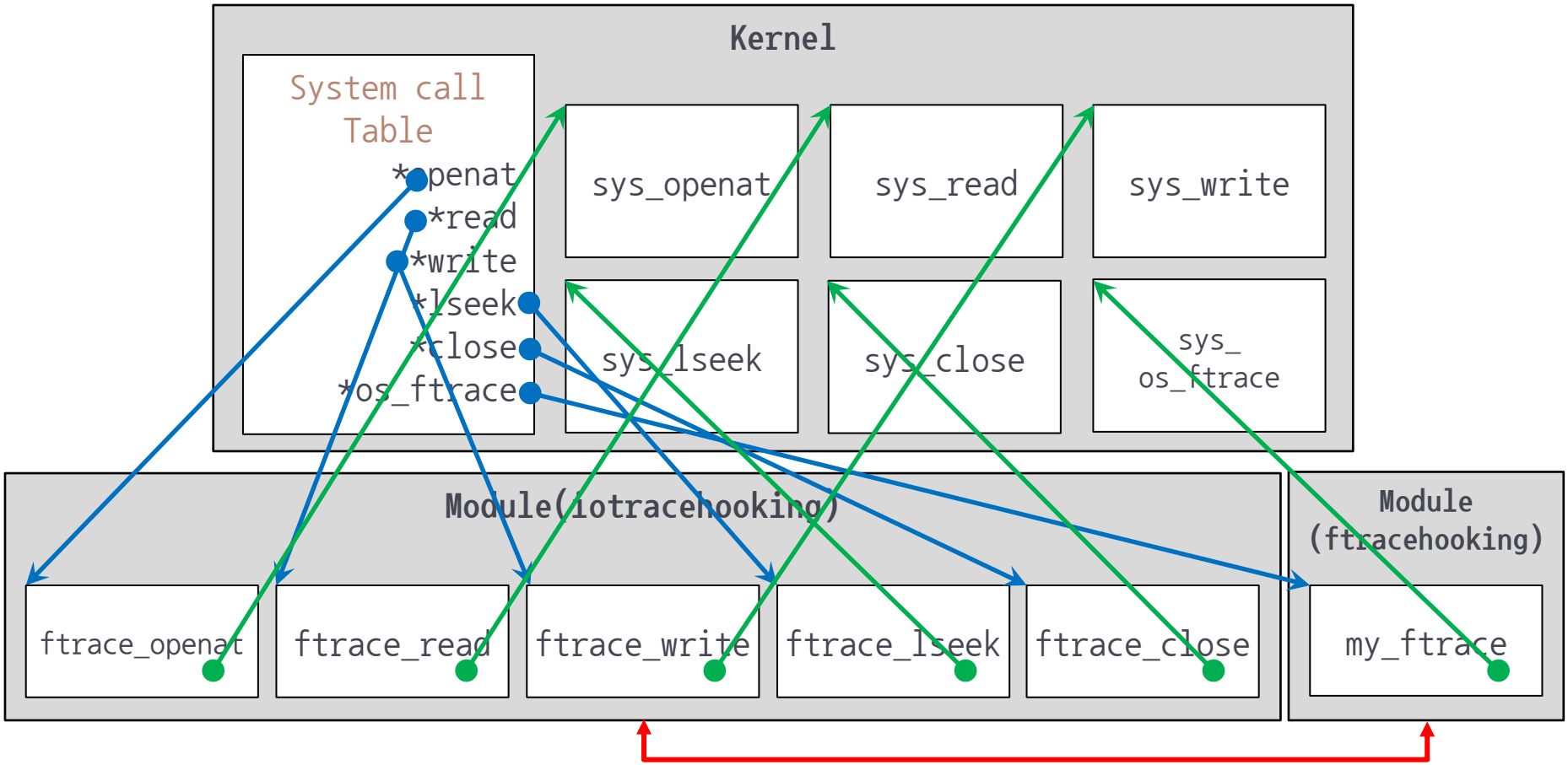
Assignment 2-3

- **Behaviors**
  - **After** inserting modules



Assignment 2-3

- **Behaviors**
  - **After** inserting modules



Assignment 2-3

- **Requirements**
  - 제출 파일
    - **1) ftracehooking.h**
      - ftracehooking.c 및 iotracehooking.c에서 사용하는 header
    - **2) ftracehooking.c (Tip : Assignment 2-2 수정하여 사용)**
      - `ftrace` 시스템콜을 hijack하여 `ftrace` 함수로 대체하는 커널 모듈 코드
      - `pid_t ftrace(pid_t pid);`
    - **3) iotracehooking.c**
      - `openat` / `read` / `write` / `lseek` / `close` 시스템콜을 hijack하여 `ftrace_openat` / `ftrace_read` / `ftrace_write` / `ftrace_lseek` / `ftrace_close` 함수로 대체하는 커널 모듈
    - **4) Makefile**
      - **ftracehooking.ko** 파일과 **iotracehooking.ko** 파일이 **동시에 생성**되도록 작성한 Makefile

Report Requirements

- **Ubuntu 20.04.6 Desktop 64bits 환경에서 채점**
- **Copy 발견 시 0점 처리**
- **보고서 구성**
  - **보고서 표지**
    - 수업 명, 과제 이름, 담당 교수님, 학번, 이름 필히 명시
      - 과제 이름 ➔ Assignment #1
  - **과제 내용**
    - Introduction
      - 과제 소개 - 4줄 이상(background 제외) 작성
    - Result
      - 수행한 내용을 캡처 및 설명
    - 고찰
      - 과제를 수행하면서 느낀 점 작성
    - Reference
      - 과제를 수행하면서 참고한 내용을 구체적으로 기록
      - 강의자료만 이용한 경우 생략 가능

Report Requirements

- **Source(총 6개)**
  - Assignment 2-1(1개)
    - os_ftrace.c
  - Assigment 2-2(1개)
    - os_ftracehooking.c
  - Assigment 2-3(4개)
    - ftracehooking.c / ftracehooking.h / iotracehooking.c / Makefile

  - 각 코드 파일은 디렉토리를 나누어서 제출
    - E.g)
      - Assignment2-1/os_ftrace.c
      - Assignment2-2/os_ftracehooking.c
      - Assignment2-3/ ftracehooking.c
        / ftarcehooking.h
        / iotracehooking.c
        / Makefile

- **Copy 발견 시 0점 처리**

KWANGWOON UNIVERSITY

Report Requirements

- **Softcopy Upload**
  - 제출 파일
    - 보고서 + 소스파일 **[하나의 압축 파일로 압축하여 제출(tar.xz)]**
    - 보고서(.pdf. 파일 변환)
    - 소스코드(**Comment 반드시 포함**)

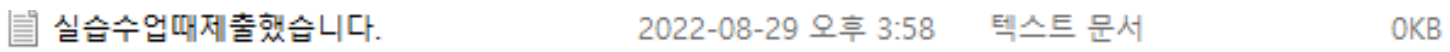
  - 보고서 및 압축 파일 명 양식
  - **OS_Assignment1 _*수강분류코드_학번_이름* 으로 작성**

수강요일	이론1 월6수5	이론2 목3	실습 금56
수강분류코드	A	B	C

    - **예시 #1)–이론(월6수5)만 수강하는 학생인 경우**
      - **보고서 OS_Assignment1_A_2024123456_홍길동.pdf**
      - **압축 파일 명: OS_Assignment1_A_2024123456_홍길동.tar.xz**
    - **예시 #2)–이론(월6수5 or 목3)과 실습 모두 수강하는 학생인 경우**
      - **보고서 OS_Assignment1_C_2024123456_홍길동.pdf**
      - **압축 파일 명: OS_Assignment1_C_2024123456_홍길동.tar.xz**
      - **+ "해당 이론반 txt 파일 제출"**

Report Requirements

- **실습 수업을 수강하는 학생인 경우**
  - **실습 과목에 과제를 제출**(.tar.xz)
  - **이론 과목에 간단한 .txt 파일**로 제출



    - **이론 과목에 .txt 파일 미 제출 시 감점**
    - **.tar.xz 파일로 제출 하지 않을 시 감점**

- **과제 제출**
  - **KLAS – 강의 과제 제출**
  - **2024년 10월 11일 금요일 23:59까지 제출**
    - **딜레이 받지 않음**
      - 제출 마감 시간 내 미제출시 해당 과제 **0점 처리(예외 없음)**