

2023년 2학기 운영체제 & 운영체제 실습

Assignment 2

System Software Laboratory

School of Computer and Information Engineering

Kwangwoon Univ.

Requirements

- **Ubuntu 16.04.5 Desktop 64bits 환경에서 채점**
- **Copy 발견 시 0점 처리**
- **보고서**
 - 표지
 - 수업 명, 과제 이름, 담당 교수님, 학번, 이름 필히 명시
 - 과제 이름 → assignment 2
 - **아래의 내용은 보고서에 필히 포함**
 - **Introduction** : 4줄 이상(background 제외) 작성
 - **Result** : 수행한 내용을 캡처 및 설명
 - **고찰** : 과제를 수행하면서 느낀점 작성
 - **Reference**
 - 과제를 수행하면서 참고한 내용을 구체적으로 기록
 - 강의자료만 이용한 경우 생략 가능

Requirements(cont'd)

- **Source**

- 아래 소스파일명과 다르게 작성하여 제출 시 감점처리됩니다.
- **ftracehooking.h**
 - ftracehooking.c 및 iotracehooking.c에서 사용하는 header
- **ftracehooking.c**
 - ftrace 시스템콜을 hijack하여 ftrace 함수로 대체하는 커널 모듈 코드
 - pid_t ftrace(pid_t pid);
- **iotracehooking.c**
 - open / read / write / lseek / close 시스템콜을 hijack하여 ftrace_open / ftrace_read / ftrace_write / ftrace_lseek / ftrace_close 함수로 대체하는 커널 모듈
- **Makefile**
 - **ftracehooking.ko** 파일과 **iotracehooking.ko** 파일이 **동시에 생성**되도록 작성한 Makefile

Requirements (cont'd)

- **Softcopy만 작성(Hardcopy 받지 않음)**
- **제출 파일**
 - 보고서(.pdf) + Source file 하나의 압축파일로 압축하여 제출(**tar.gz**)
 - Tar 압축 및 해제 방법
 - 압축 시 → `tar -zcvf [압축 파일명].tar.gz[폴더 명]`
 - 해제 시 → `tar -zxvf 파일명.tar.gz`

- **보고서 및 압축 파일 명 양식**

- **os_과제번호_학번_수강분류코드**


수강요일	이론_화목	이론_금12	실습
수강분류코드	A	B	C

- e.g.
 - 이론_월수 수강하는 학생인 경우
(보고서) **os_2_20231234567_A.pdf** (압축파일) **os_2_20231234567_A.tar.gz**
 - 이론_금12 수강하는 학생인 경우
(보고서) **os_2_20231234567_B.pdf** (압축파일) **os_2_20231234567_B.tar.gz**
 - 실습 수강하는 학생인 경우
(보고서) **os_2_20231234567_C.pdf** (압축파일) **os_2_20231234567_C.tar.gz**

Requirements (cont'd)

- 실습 수업을 수강하는 학생인 경우

- 실습 과목에 과제를 제출(.tar.gz)
- 이론 과목에 간단한 .txt 파일로 제출

 실습수업때제출했습니다.

2022-08-29 오후 3:58 텍스트 문서

OKB

- 이론 과목에 .txt 파일 미 제출 시 감점

- 과제 제출

- KLAS – 강의 과제 제출
- 2023년 10월 12일 목요일 23:59까지 제출
- 딜레이 받지 않음
 - 제출 마감 시간 내 미제출시 해당 과제 0점 처리

Assignment 2

- **ftrace : File Tracing**

- 특정 pid에 대하여 **파일에 관한 시스템콜을 추적**하는 툴 작성
- 결과 예시

```
[ 250.555709] OS Assignment2 ftrace [2692] Start  
[ 250.555726] [SSLAB] /a.out file[abc.txt] stats [x] read - 20 / written - 26  
[ 250.555726] open[1] close[1] read[4] write[5] lseek[9]  
[ 250.555726] OS Assignment2 ftrace [2692] End
```

- 추적할 시스템콜
 - open / read / write / lseek / close

Assignment 2

■ Requirements

- 시스템콜
 - 시스템콜 명 : ftrace
 - 시스템콜 번호 : 336번
 - 커널 소스 폴더에 ftrace 폴더를 생성후 진행
 - `asmlinkage int ftrace(pid_t pid);`
- 커널 log 메시지 (결과 예시 참고)

```
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 Assignment 2 ftrace [pid] End
```

Assignment 2

■ Requirements

- Trace 시작
 - ftrace 시스템콜에 특정 pid 값을 넣고 호출
 - 커널 메시지
 - OS Assignment 2 ftrace [pid] Start
- Trace 종료
 - Trace 종료 시점 : ftrace 시스템콜에 0 값을 넣고 호출
 - [학번] /**process_name** file[**file_name**] stats [x] read - **#_read_bytes** / written - **#_written_bytes**
 - [시간] open[#_open 횟수] close[#_close 횟수] read[#_read 횟수] write[#_write 횟수] lseek[#_lseek 횟수]
 - trace 종료 시점 : ftrace 시스템콜에 0 값을 넣고 호출

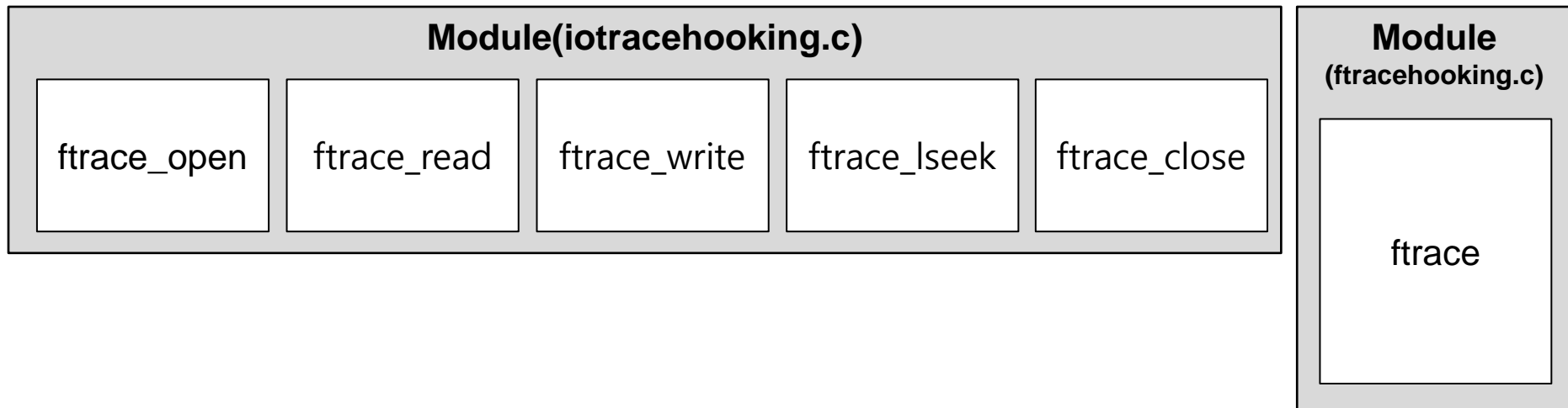
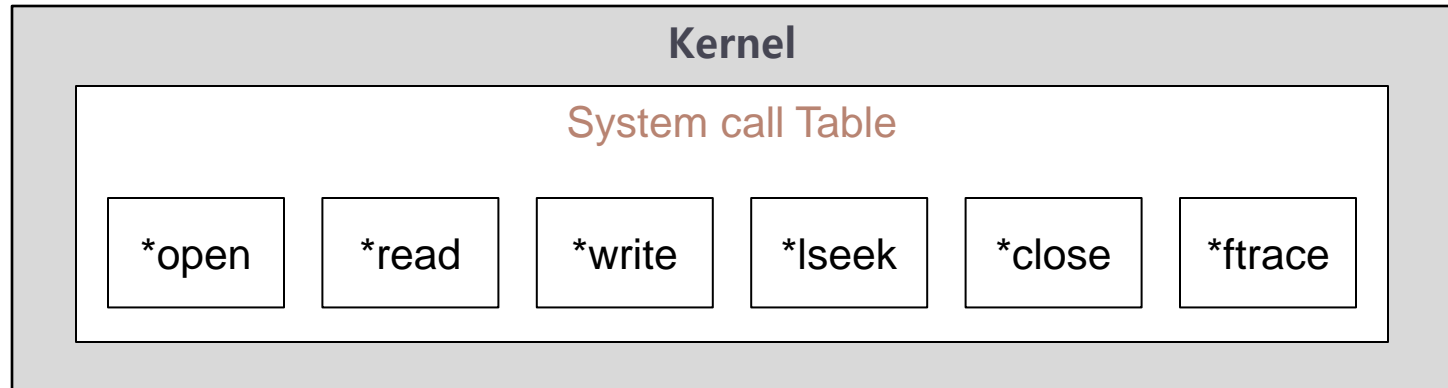
Assignment 2

▪ Step by step

- ftrace 시스템콜 "ftrace"을 만든다.(시스템콜 번호: 336번)
 - PID의 인자를 받는 시스템콜
- ftrace 시스템콜을 hijack 하여 ftrace 함수로 대체한다.
 - ftracehooking.c
 - static asmlinkage int ftrace(const struct pt_regs *regs) 사용
 - iotracehooking.c와 연동
 - EXPORT_SYMBOL() 사용
- open, read, write, lseek, close 시스템콜의 원형을 찾는다.
- open / read / write / lseek / close 시스템콜을 hijack하여 ftrace_open / ftrace_read / ftrace_write / ftrace_lseek / ftrace_close 함수로 대체한다.
 - iotracehooking.c
 - ex. static asmlinkage long ftrace_open(const struct pt_regs *regs) 사용
- 테스트용 프로그램 작성
 - .txt 파일 생성 후 open/read/write/lseek/close 사용하는 프로그램
 - 강의자료실에 첨부한 'test.c' 프로그램 참고 및 사용 가능

Assignment 2

- Behaviors



Assignment 2

- Behaviors

