

실습 과제2

시스템콜 작동 과정 분석

1. C 라이브러리 래퍼 함수 호출

- **설명:** 사용자가 `write(fd, buf, count)` 와 같은 함수 호출 → 실제로는 glibc에서 제공하는 래퍼 함수를 호출하는 것
- **역할:** 내부적으로 적절한 시스템 콜 번호와 인자 준비하여 커널에 진입하는 역할 수행
- **목적:** 사용자 코드와 커널 시스템 콜 인터페이스 사이의 추상화 계층을 제공 (사용자 → syscall 준비)

2. `syscall` 명령어를 통한 커널 모드 진입

- **설명:** 래퍼 함수는 `syscall` 어셈블리 명령어를 사용하여 사용자 모드 → 커널 모드로 진입
- **역할:** CPU가 모드 전환을 하여, 시스템 콜을 처리할 수 있게 함
- **목적:** 사용자 프로그램이 직접 하드웨어나 시스템 리소스에 접근하면 보안 위험과 시스템 불안정성이 생기기 때문에 커널이 모든 요청을 통제하고 검사하도록 함

3. 시스템 콜 번호 전달

- **설명:** `write` 의 시스템 콜 번호는 1번이며, 레지스터를 통해 함께 전달됨
- **역할:** 커널이 어떤 시스템 콜을 요청받았는지 식별할 수 있게 해주는 ID 역할
- **목적:** 수많은 시스템 콜 중 어떤 기능을 수행할 지를 결정하는 데 사용됨

4. 시스템 콜 테이블에서 함수 검색

- **설명:** 커널은 전달받은 시스템 콜 번호(예: 1번)를 바탕으로 `syscall_64.tbl` 또는 내부 시스템 콜 테이블에서 해당 번호에 매핑된 함수 포인터 탐색
- **역할:** 해당 시스템 콜에 연결된 핸들러 함수 결정
- **목적:** 시스템 콜 번호 → 실제 커널 함수로 매핑하는 메커니즘 제공

5. 커널 핸들러 함수 실행

- **설명:** `sys_write()` 라는 커널 내부 함수가 실행되어, 해당 파일 디스크립터에 데이터를 출력하는 작업 수행
- **역할:** 커널에서 실제 기능 수행 (ex. `fd=1` 이면 콘솔 출력 (표준 출력))

- **목적:** 사용자 요청에 따라 커널 리소스를 안전하게 조작

6. 결과값 설정 및 사용자 모드로 복귀 (`sysret`)

- **설명:** `sys_write()` 가 실행을 마치면, 반환 값을 `rax` 레지스터에 저장하고, `sysret` 명령어를 통해 다시 사용자 모드로 전환
- **역할:** 커널 모드 → 사용자 모드 복귀 및 리턴값 전달
- **목적:** 시스템 콜 호출자에게 결과 전달 (ex. 출력한 바이트 수)

[User Program]

```
|
| 호출: write(fd, buf, count)
|   - fd = 1 (stdout)
|   - buf = "Hello\n"
|   - count = 6
|
↓
```

```
| syscall 명령어 실행          |
| - rax = 1 (write 시스템 콜 번호) |
| - rdi = fd                  |
| - rsi = buf 포인터          |
| - rdx = count                |
|
```

↓

[Kernel Mode]

```
|
| 커널이 시스템 콜 번호(rax = 1)를 기준으로
| syscall 테이블에서 sys_write 함수 호출
|
| sys_write(fd, buf, count)
|   → buf 주소의 메모리에서 데이터 복사 ("Hello\n")
|   → fd가 가리키는 파일(예: stdout)에 출력
|
↓
결과: 성공적으로 6바이트 출력됨
```

```
| sysret 명령어 실행          |
| - rax = 6 (출력한 바이트 수) |
|
```



[User Program 복귀]



| write() 함수가 6을 리턴함

| → 출력 성공

시스템콜 생성 및 커널 컴파일 실습

- 시스템 콜 테이블(`syscall_64.tbl` or `unistd.h`)에 새로운 시스템 콜을 등록한 스크린샷

```
GNU nano 7.2 /usr/src/linux-source-6.8.0/arch/x86/entry/syscalls/syscall_64.tbl
427 common io_uring_register sys_io_uring_register
428 common open_tree sys_open_tree
429 common move_mount sys_move_mount
430 common fsopen sys_fsopen
431 common fsconfig sys_fsconfig
432 common fsmount sys_fsmount
433 common fspick sys_fspick
434 common pidfd_open sys_pidfd_open
435 common clone3 sys_clone3
436 common close_range sys_close_range
437 common openat2 sys_openat2
438 common pidfd_getfd sys_pidfd_getfd
439 common faccessat2 sys_faccessat2
440 common process_madvise sys_process_madvise
441 common epoll_pwait2 sys_epoll_pwait2
442 common mount_setattr sys_mount_setattr
443 common quotactl_fd sys_quotactl_fd
444 common landlock_create_ruleset sys_landlock_create_ruleset
445 common landlock_add_rule sys_landlock_add_rule
446 common landlock_restrict_self sys_landlock_restrict_self
447 common memfd_secret sys_memfd_secret
448 common process_mrelease sys_process_mrelease
449 common futex_waitv sys_futex_waitv
450 common set_mempolicy_home_node sys_set_mempolicy_home_node
451 common cachestat sys_cachestat
452 common fchmodat2 sys_fchmodat2
453 64 map_shadow_stack sys_map_shadow_stack
454 common futex_wake sys_futex_wake
455 common futex_wait sys_futex_wait
456 common futex_requeue sys_futex_requeue
457 common statmount sys_statmount
458 common listmount sys_listmount
459 common lsm_get_self_attr sys_lsm_get_self_attr
460 common lsm_set_self_attr sys_lsm_set_self_attr
461 common lsm_list_modules sys_lsm_list_modules
462 common print_student_name sys_print_student_name
463 common print_student_id sys_print_student_id
464 common print_student_info sys_print_student_info
#
# Due to a historical design error, certain syscalls are numbered differently
# in x32 as compared to native x86_64. These syscalls have numbers 512-547.
# Do not add new syscalls to this range. Numbers 548 and above are available
# for non-x32 use.
#
512 x32 rt_sigaction compat_sys_rt_sigaction
513 x32 rt_sigreturn compat_sys_x32_rt_sigreturn
```

462	common	print_student_name	sys_print_student_name
463	common	print_student_id	sys_print_student_id
464	common	print_student_info	sys_print_student_info

- 시스템 콜 번호, 함수 매핑

- 462 → `sys_print_student_name`
- 463 → `sys_print_student_id`
- 464 → `sys_print_student_info`

- 필요성

- 사용자 공간에서 `syscall(번호, print_student...)` 을 호출했을 때, 커널은 이 테이블을 참조해 해당 번호가 어떤 C 함수(`sys_*`)로 연결되는지 판단
- 매핑이 없으면 “unimplemented syscall(-ENOSYS)” 에러 발생
⇒ 번호 → 함수 연결을 반드시 등록해야 새 시스템 콜이 정상 동작함
- `syscalls.h` 에 새로운 시스템 콜을 등록한 스크린샷

```

GNU nano 1.2 /usr/src/linux-source-6.8.0/include/linux/syscalls.h
/* for __ARCH_WANT_SYS_IPC */
long ksys_semop(int semid, struct sembuf __user *tsems,
               unsigned int nsops,
               const struct __kernel_timespec __user *timeout);
long ksys_semget(key_t key, int nsems, int semflg);
long ksys_old_semctl(int semid, int semnum, int cmd, unsigned long arg);
long ksys_msgsget(key_t key, int msgflg);
long ksys_msgctl(int msqid, int cmd, struct msqid_ds __user *buf);
long ksys_msgrcv(int msqid, struct msgbuf __user *msgp, size_t msgsz,
               long msgtyp, int msgflg);
long ksys_msgsnd(int msqid, struct msgbuf __user *msgp, size_t msgsz,
               int msgflg);
long ksys_shmget(key_t key, size_t size, int shmflg);
long ksys_shmctl(char __user *shmaddr);
long ksys_old_shmctl(int shmid, int cmd, struct shmid_ds __user *buf);
long compat_ksys_semop(int semid, struct sembuf __user *tsems,
               unsigned int nsops,
               const struct old_timespec32 __user *timeout);
long __do_semop(int semid, struct sembuf *tsems, unsigned int nsops,
               const struct timespec64 *timeout,
               struct ipc_namespace *ns);

int __sys_getsockopt(int fd, int level, int optname, char __user *optval,
               int __user *optlen);
int __sys_setsockopt(int fd, int level, int optname, char __user *optval,
               int optlen);
asmlinkage void sys_print_student_name(char *name);
asmlinkage void sys_print_student_id(char *id);
asmlinkage void sys_print_student_info(char *school, char *major);
#endif
  
```

```

#ifdef CONFIG_X86_64
...
asmlinkage void sys_print_student_name(char *name);
asmlinkage void sys_print_student_id(char *id);
asmlinkage void sys_print_student_info(char *school, char *major);
#endif
  
```

○ 추가한 프로토타입

- `sys_print_student_name(char *name)`
- `sys_print_student_id(char *id)`
- `sys_print_student_info(char *school, char *major)`

○ 헤더 파일 수정 필요성

1. 컴파일러에게 시그니처(인자 타입·반환 타입) 알려주기

- 선언 없이는 암묵적 선언(implicit declaration) 오류 발생 → 빌드 중단

2. 링크 단계에서 심볼 해석

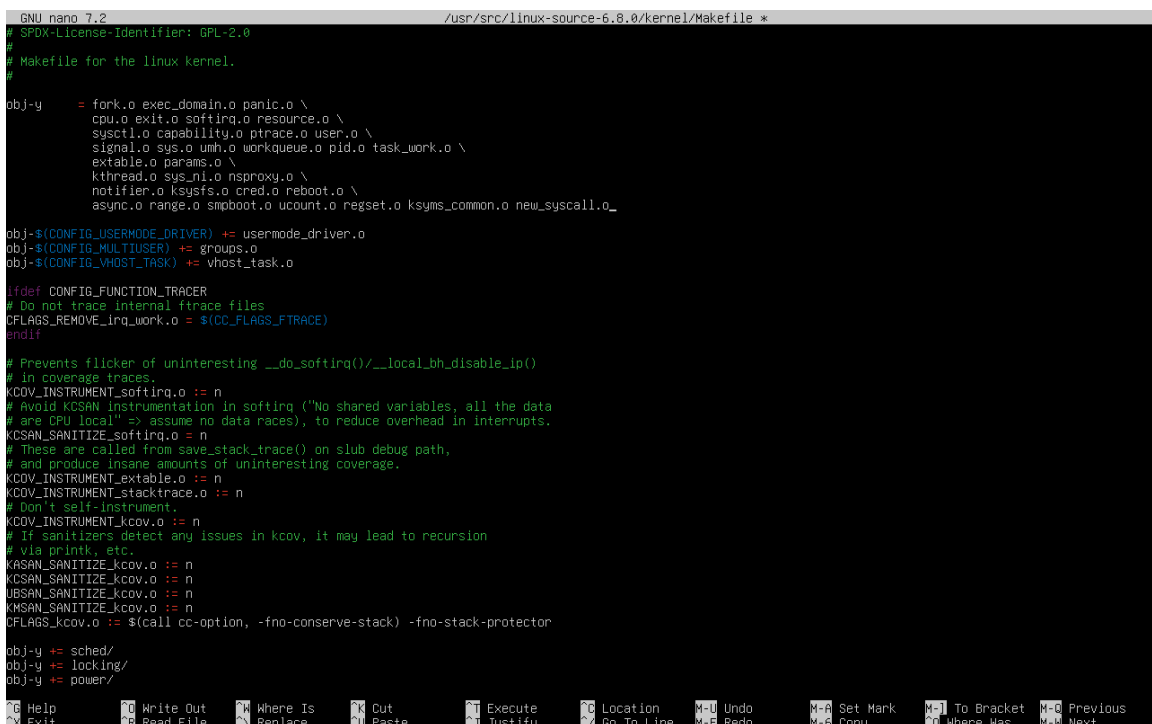
- 커널이 `syscall_64.tbl` 에 매핑된 함수명을 찾을 때, 프로토타입이 없으면 정의된 함수와 바인딩할 수 없음

3. `asmlinkage` 로 호출 규약 지정

- 시스템 콜은 모든 인자를 스택에서 받도록 설계해야 하므로, 올바른 ABI를 강제함

⇒ 앞 단계에서는 `sys_print_*` 함수들을 **정의(definition)**한 것이고, 이 단계에서는 다른 커널 코드(빌드 시스템)가 이 함수들을 인식할 수 있도록 함수 시그니처를 알려주는 **선언(declaration)**을 추가한 것

- `Makefile` 에 새로운 시스템 콜 함수 파일을 추가한 스크린샷



```
GNU nano 7.2 /usr/src/linux-source-6.8.0/kernel/Makefile *
# SPDX-License-Identifier: GPL-2.0
#
# Makefile for the linux kernel.
#
obj-y += fork.o exec_domain.o panic.o \
        cpu.o exit.o softirq.o resource.o \
        sysctl.o capability.o ptrace.o user.o \
        signal.o sys.o umh.o workqueue.o pid.o task_work.o \
        extable.o params.o \
        kthread.o sys_ni.o nsproxy.o \
        notifier.o ksysfs.o cred.o reboot.o \
        async.o range.o smpboot.o ucount.o regset.o ksyms_common.o new_syscall.o

obj-$(CONFIG_USERMODE_DRIVER) += usermode_driver.o
obj-$(CONFIG_MULTIUSER) += groups.o
obj-$(CONFIG_VHOST_TASK) += vhost_task.o

ifdef CONFIG_FUNCTION_TRACER
# Do not trace internal ftrace files
CFLAGS_REMOVE_irq_work.o = $(CC_FLAGS_FTRACE)
endif

# Prevents flicker of uninteresting __do_softirq()/__local_bh_disable_ip()
# in coverage traces.
KCOV_INSTRUMENT_softirq.o := n
# Avoid KCSAN instrumentation in softirq ("No shared variables, all the data
# are CPU local" => assume no data races), to reduce overhead in interrupts.
KCSAN_SANITIZE_softirq.o := n
# These are called from save_stack_trace() on slub debug path,
# and produce insane amounts of uninteresting coverage.
KCOV_INSTRUMENT_extable.o := n
KCOV_INSTRUMENT_stacktrace.o := n
# Don't self-instrument.
KCOV_INSTRUMENT_kcov.o := n
# If sanitizers detect any issues in kcov, it may lead to recursion
# via printk, etc.
KCSAN_SANITIZE_kcov.o := n
KCSAN_SANITIZE_kcov.o := n
UBSAN_SANITIZE_kcov.o := n
KMSAN_SANITIZE_kcov.o := n
CFLAGS_kcov.o := $(call cc-option, -fno-conserve-stack) -fno-stack-protector

obj-y += sched/
obj-y += locking/
obj-y += power/
```

◦ 추가된 내용

`ksyms_common.o` 뒤에 `new_syscall.o` 를 추가함

```
obj-y += ... ksyms_common.o new_syscall.o
```

`new_syscall.o` 가 `obj-y` (정적 링크될 오브젝트 목록)에 포함되도록 함

○ 커널 빌드에 미치는 영향

1. 컴파일 단계

- kbuild가 `new_syscall.c` 를 찾아 `new_syscall.o` 로 컴파일
(`obj-y` 에 없으면 해당 소스 파일은 컴파일 대상에서 빠짐)

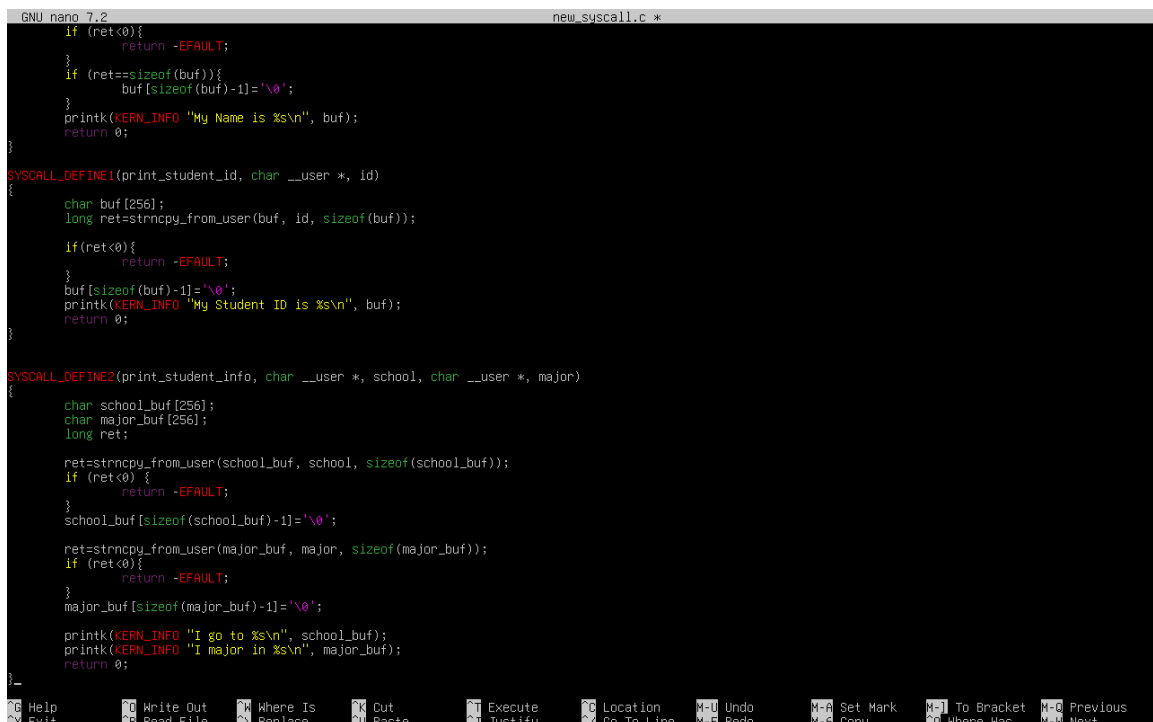
2. 링크 단계

- 생성된 `new_syscall.o` 가 `vmlinux` (커널 이미지)에 정적으로 링크되어, 새로 정의한 시스템 콜 함수가 커널 바이너리에 포함됨

3. 결과

- 이 설정이 없으면 `new_syscall.c` 정의 함수 빌드에 반영X → “undefined reference” 또는 “unimplemented syscall” 오류 발생
- 설정 추가 후에는 커널에 함수가 정상 탑재 → `syscall(번호, ...)` 호출이 실제 구현으로 분기

• 작성한 `new_syscall.c` 스크린샷



```
GNU nano 1.2 new_syscall.c *
if (ret<0){
    return -EFAULT;
}
if (ret==sizeof(buf)){
    buf[sizeof(buf)-1]='\0';
}
printk(KERN_INFO "My Name is %s\n", buf);
return 0;
}

SYSCALL_DEFINE1(print_student_id, char __user *, id)
{
    char buf[256];
    long ret=strncpy_from_user(buf, id, sizeof(buf));

    if(ret<0){
        return -EFAULT;
    }
    buf[sizeof(buf)-1]='\0';
    printk(KERN_INFO "My Student ID is %s\n", buf);
    return 0;
}

SYSCALL_DEFINE2(print_student_info, char __user *, school, char __user *, major)
{
    char school_buf[256];
    char major_buf[256];
    long ret;

    ret=strncpy_from_user(school_buf, school, sizeof(school_buf));
    if (ret<0) {
        return -EFAULT;
    }
    school_buf[sizeof(school_buf)-1]='\0';
    ret=strncpy_from_user(major_buf, major, sizeof(major_buf));
    if (ret<0){
        return -EFAULT;
    }
    major_buf[sizeof(major_buf)-1]='\0';

    printk(KERN_INFO "I go to %s\n", school_buf);
    printk(KERN_INFO "I major in %s\n", major_buf);
    return 0;
}

_
G Help      C Write Out  W Where Is   C Cut        E Execute    C Location  M Undo      M Set Mark  M To Bracket M Previous  M Next
X Exit      R Read File  R Replace   V Paste      J Justify   G Go To Line M Redo     M Copy      Q Where Was  M Next
```

1. `print_student_name`

- 인자:
 - `char __user *name` : 사용자 공간에 저장된 학생 이름 문자열의 포인터
- 동작:

1. `strncpy_from_user(buf, name, 256)` 로 사용자 공간의 `name` 을 커널 버퍼 `buf[256]` 으로 복사
2. 복사 실패 시 `EFAULT` 반환
3. 복사 길이가 버퍼 크기와 같으면 마지막 바이트를 `'\0'` 로 강제 널 종단
4. `printk(KERN_INFO "My Name is %s\n", buf)` 로 `dmesg` 로그에 "My Name is ..." 형태로 출력
5. 성공 시 `0` 반환

2. `print_student_id`

- 인자:
 - `char __user *id` : 사용자 공간에 저장된 학생 학번 문자열의 포인터
- 동작:
 1. `strncpy_from_user(buf, id, 256)` 로 `id` 를 `buf` 에 복사
 2. 복사 실패 시 `EFAULT` 반환
 3. 마지막 바이트를 무조건 `'\0'` 로 설정하여 널 종단 보장
 4. `printk(KERN_INFO "My Student ID is %s\n", buf)` 로 "My Student ID is ..." 형태로 출력
 5. 성공 시 `0` 반환

3. `print_student_info`

- 인자:
 - `char __user *school` : 사용자 공간의 학교 이름 문자열 포인터
 - `char __user *major` : 사용자 공간의 전공명 문자열 포인터
- 동작:
 1. `strncpy_from_user(school_buf, school, 256)` 로 학교 이름을 `school_buf` 에 복사
 2. 복사 실패 시 `EFAULT` 반환; 널 종단 보장
 3. `strncpy_from_user(major_buf, major, 256)` 로 전공명을 `major_buf` 에 복사
 4. 복사 실패 시 `EFAULT` 반환; 널 종단 보장
 5. `printk(KERN_INFO "I go to %s\n", school_buf)` 로 "I go to ..." 출력
 6. `printk(KERN_INFO "I major in %s\n", major_buf)` 로 "I major in ..." 출력

7. 성공 시 0 반환

⇒ 각 함수는 사용자 공간 문자열을 안전하게 커널로 복사한 뒤, `printk(KERN_INFO, ...)` 를 이용해 커널 로그에 해당 정보를 출력하도록 구현

- **.config** 파일 수정한 내용이 포함된 스크린샷

```
GNU nano 7.2                                .config *
CONFIG_CRYPTODEV_DEV_IAA_CRYPTOM=m
# CONFIG_CRYPTODEV_DEV_IAA_CRYPTOSTATS is not set
CONFIG_CRYPTODEV_DEV_CHELSIO=m
CONFIG_CRYPTODEV_DEV_VIRTIO=m
CONFIG_CRYPTODEV_DEV_SAFEXCEL=m
CONFIG_CRYPTODEV_DEV_AMLOGIC_GXL=m
# CONFIG_CRYPTODEV_DEV_AMLOGIC_GXL_DEBUG is not set
CONFIG_ASYMMETRIC_KEY_TYPE=y
CONFIG_ASYMMETRIC_PUBLIC_KEY_SUBTYPE=y
CONFIG_X509_CERTIFICATE_PARSER=y
CONFIG_PKCS8_PRIVATE_KEY_PARSER=m
CONFIG_PKCS7_MESSAGE_PARSER=y
CONFIG_PKCS7_TEST_KEY=m
CONFIG_SIGNED_PE_FILE_VERIFICATION=y
# CONFIG_FIPS_SIGNATURE_SELFTEST is not set

#
# Certificates for signature checking
#
CONFIG_MODULE_SIG_KEY="certs/signing_key.pem"
CONFIG_MODULE_SIG_KEY_TYPE_RSA=y
# CONFIG_MODULE_SIG_KEY_TYPE_ECDSA is not set
CONFIG_SYSTEM_TRUSTED_KEYRING=y
CONFIG_SYSTEM_TRUSTED_KEYS=""
CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
CONFIG_SECONDARY_TRUSTED_KEYRING=y
# CONFIG_SECONDARY_TRUSTED_KEYRING_SIGNED_BY_BUILTIN is not set
CONFIG_SYSTEM_BLACKLIST_KEYRING=y
CONFIG_SYSTEM_BLACKLIST_HASH_LIST=""
CONFIG_SYSTEM_REVOCATION_LIST=y
CONFIG_SYSTEM_REVOCATION_KEYS=""
# CONFIG_SYSTEM_BLACKLIST_AUTH_UPDATE is not set
# end of Certificates for signature checking

CONFIG_BINARY_PRINTF=y

#
# Library routines
#
CONFIG_RAID6_PQ=m
CONFIG_RAID6_PQ_BENCHMARK=y
CONFIG_LINEAR_RANGES=y
CONFIG_PACKING=y
CONFIG_BITREVERSE=y
CONFIG_GENERIC_STRNCPY_FROM_USER=y

? Help      ? Write Out  ? Where Is   ? Cut        ? Execute    ? Location   ? Undo       ? Set Mark   ? To Bracket  ? Previous
? Exit      ? Read File  ? Replace    ? Paste      ? Interrupt  ? Go To Line ? Endo       ? Ser        ? Where Mac  ? Wwst
```

- 설정 변경: 시스템 키 파일 경로를 가리키는 설정을 빈 문자열로 바꿈

- 역할 및 변경 이유

- CONFIG_SYSTEM_TRUSTED_KEYS

- 커널과 모듈 서명 검증에 사용할 공개키 파일 경로 리스트
- 기본값이 실제 파일을 가리키면, 그 파일이 없을 때 빌드/부팅 중 "파일 없음" 오류 발생
 - ⇒ 빈 문자열로 두면 "검증용 키 파일이 없다"는 에러를 피할 수 있음

- CONFIG_SYSTEM_REVOCATION_KEYS

- 모듈 서명 해제(취소) 처리를 위한 키 파일 경로 리스트
 - 기본값이 실제 파일을 가리키면, 그 파일이 없을 때 빌드/부팅 중 "파일 없음" 오류 발생
- ⇒ 빈 문자열로 두면 "취소 키 파일 없음" 에러를 피할 수 있음

⇒ 미리 생성/관리되지 않은 서명 키 때문에 커널 빌드나 부팅이 중단되는 것을 막고자 이 두 경로의 문자열을 빈 문자열로 변경

- 커널 컴파일 후, 커널 버전을 확인할 수 있는 스크린샷

```
Ubuntu 24.04.2 LTS 2021250146 tty1
2021250146 login: jsm
Password:
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.12 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Thu Apr 17 01:26:16 PM UTC 2025

System load:          0.37
Usage of /:            49.0% of 76.60GB
Memory usage:         4%
Swap usage:           0%
Processes:            101
Users logged in:      0
IPv4 address for enp0s3: 10.0.2.15
IPv6 address for enp0s3: fd00::a00:27ff:fe71:b221

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.
   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

4 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

jsm@2021250146:~$ uname -r
6.8.12
jsm@2021250146:~$ _
```

- 현재 실행 중인 커널의 버전 출력

⇒ 새로 컴파일/설치한 커널이 정상적으로 OS를 구동 중임을 확인할 수 있음

- 작성한 `call_new_syscall.c` 스크린샷

```
GNU nano 7.2 call_new_syscall.c
#include <unistd.h>

int main(void)
{
    char name[]="Sumin Jung";
    char id[]="2021250146";
    char school[]="Korea University";
    char major[]="Biosystem and Biomedical Science";
    syscall(462, name);
    syscall(463, id);
    syscall(464, school, major);
    return 0;
}
```

○ 헤더

```
#include <unistd.h>
```

⇒ `syscall(long number, ...)` 함수 프로토타입을 가져옴

○ 인자 문자열 준비

```
char name[] = "Sumin Jung";
char id[] = "2021250146";
char school[] = "Korea University";
char major[] = "Biosystem and Biomedical Science";
```

○ `syscall()` 호출

```
syscall(462, name);
syscall(463, id);
syscall(464, school, major);
```

- 첫 번째 파라미터(462, 463, 464)는 시스템 콜 테이블에 등록된 번호
- 그 뒤 인자들은 각 시스템 콜이 선언된 프로토타입(`print_student_name(name)` , `print_student_id(id)` , `print_student_info(school, major)`)에 대응됨

앞에 정의한 인자 문자열을 각각 함수에 인자로 넘겨줌

- 실행 시 커널은 이 번호를 보고 `sys_print_student_name`, `sys_print_student_id`, `sys_print_student_info` 함수를 각각 호출함
- `call_new_syscall.c` 컴파일 및 실행 후, `dmesg` 를 사용한 커널 메시지 출력 스크린샷

```
13.016065] vmwgfx 0000:00:02.0: [drm] *ERROR* Please switch to a supported graphics device to avoid problems.
13.016073] vmwgfx 0000:00:02.0: [drm] DMA map mode: Caching DMA mappings.
13.016141] vmwgfx 0000:00:02.0: [drm] Legacy memory limits: VRAM = 16384 kB, FIFO = 2048 kB, surface = 507904 kB
13.016144] vmwgfx 0000:00:02.0: [drm] MOB limits: max mob size = 131072 kB, max mob pages = 4096
13.016147] vmwgfx 0000:00:02.0: [drm] Max GMR ids is 8192
13.016148] vmwgfx 0000:00:02.0: [drm] Max number of GMR pages is 1048576
13.016150] vmwgfx 0000:00:02.0: [drm] Maximum display memory size is 16384 kB
13.035590] vmwgfx 0000:00:02.0: [drm] Screen Target display unit initialized
13.035945] vmwgfx 0000:00:02.0: [drm] Fifo max 0x00200000 min 0x00001000 cap 0x00000355
13.036510] vmwgfx 0000:00:02.0: [drm] Using command buffers with DMA pool.
13.036521] vmwgfx 0000:00:02.0: [drm] Available shader model: Legacy.
13.041202] EXT4-fs (sda2): mounted filesystem 8734b479-4333-4be7-b02b-9f92954b93be r/w with ordered data mode. Quota mode: none.
13.043467] [drm] Initialized vmwgfx 2.20.0 20211206 for 0000:00:02.0 on minor 0
13.045405] fbcon: vmwgfxdrmfb (fb0) is primary device
13.056273] Console: switching to colour frame buffer device 160x50
13.059503] vmwgfx 0000:00:02.0: [drm] fb: vmwgfxdrmfb frame buffer device
13.209908] audit: type=1400 audit(1744896351.910:2): apparmor="STATUS" operation="profile_load" profile="unconfined" name="ipassword" pid=507 comm="apparmor_parser"
13.214278] snd_intel8x0 0000:00:05.0: allow list rate for 1028:0177 is 48000
13.220750] audit: type=1400 audit(1744896351.921:3): apparmor="STATUS" operation="profile_load" profile="unconfined" name="Discord" pid=512 comm="apparmor_parser"
13.229602] audit: type=1400 audit(1744896351.931:4): apparmor="STATUS" operation="profile_load" profile="unconfined" name="406F6E676F44220436F6D70617373 pid=515 comm="apparmor_parser"
13.237458] audit: type=1400 audit(1744896351.939:5): apparmor="STATUS" operation="profile_load" profile="unconfined" name="QtWebEngineProcess" pid=517 comm="apparmor_parser"
13.245065] audit: type=1400 audit(1744896351.947:6): apparmor="STATUS" operation="profile_load" profile="unconfined" name="balena-etcher" pid=520 comm="apparmor_parser"
13.253066] audit: type=1400 audit(1744896351.955:7): apparmor="STATUS" operation="profile_load" profile="unconfined" name="brave" pid=523 comm="apparmor_parser"
13.260494] audit: type=1400 audit(1744896351.962:8): apparmor="STATUS" operation="profile_load" profile="unconfined" name="buildah" pid=525 comm="apparmor_parser"
13.268718] audit: type=1400 audit(1744896351.971:9): apparmor="STATUS" operation="profile_load" profile="unconfined" name="busybox" pid=527 comm="apparmor_parser"
13.277496] audit: type=1400 audit(1744896351.979:10): apparmor="STATUS" operation="profile_load" profile="unconfined" name="cam" pid=529 comm="apparmor_parser"
13.284491] audit: type=1400 audit(1744896351.986:11): apparmor="STATUS" operation="profile_load" profile="unconfined" name="ch-checkns" pid=531 comm="apparmor_parser"
13.696858] cfg80211: Loading compiled-in X.509 certificates for regulatory database
13.697570] Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
13.697703] Loaded X.509 cert 'wens: 61c038651aabdcf94bd0ac7ff06c7248db18c600'
13.754996] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
15.060939] loop0: detected capacity change from 0 to 8
15.226241] NET: Registered PF_QIPCRTR protocol family
38.374763] systemd-journal[297]: File /var/log/journal/2966b1db1c164e86a1f503e64cbaaa03/user-1000.journal corrupted or uncleanly shut down, renaming and replacing.
476.062354] My Name is Sumin Jung
476.062361] My Student ID is 2021250146
476.062363] I go to Korea University
476.062365] I major in Biosystem and Biomedical Science
ism@2021250146: ~/workspace$
```

- `dmesg` 출력 맨 아래의 문장으로 시스템 콜이 성공적으로 작동했음을 확인할 수 있음

- 각 행의 메시지 내용이 `printk(KERN_INFO ...)` 호출부와 정확히 일치
- 시스템 콜 호출 번호(462, 463, 464)에 대응하는 함수들이 순서대로 실행되어 로그 찍음
- 에러 코드 없이 0 반환 (프로그램 상에서 에러 체크를 하지 않아도 로그가 찍혔다는 자체가 정상 수행의 증거)