

# 운영체제 실습 2 과제

2021320308 박한준

## 시스템콜 작동 과정 분석

### 1. 사용자 공간에서의 호출

#### a. 과정

- 사용자 프로그램에서 C 라이브러리 함수 `printf()` 가 호출된다.
- glibc에서 제공하는 C 라이브러리 래퍼 함수 `write(fd, *buf, count)` 가 호출된다.

이 함수는 시스템 콜 인터페이스에게 시스템 콜 식별 번호(`write()`의 경우 1번), 파일 디스크립터, 버퍼 주소 등을 전달한다.

#### b. 목적

- 복잡한 시스템 콜을, 래퍼함수를 통해 추상화하여, 유저 친화적인 고수준 프로그래밍 언어로 편리하게 호출한다.

### 2. 커널 모드로 전환

#### a. 과정

- 사용자 프로그램의 실행 상태(context)는 return 될 때를 위해 저장되고, 커널 코드 실행을 위한 환경을 준비한다.
- 시스템 콜 인터페이스를 통해 `syscall` 이 실행되고, CPU는 즉시 유저 모드에서 커널 모드로 전환한다. 미리 정해진 커널의 진입점부터 실행이 시작된다.

#### b. 목적

- 유저 단과 커널 단의 안전한 전환과 권한 변경을 담당한다.

### 3. 커널 내부 처리

#### a. 과정

- 요청 내용의 번호(1번)와 매핑되는 함수를 "시스템 콜 테이블"에서 찾고, 핸들러 함수가 그 시스템 콜의 핵심 로직을 수행한다. 수행이 끝나면 핸들러 함수는 return값을 준비한다.

#### b. 목적

- 커널 단에서 빠르고 안전하게 시스템 콜을 처리한다.

### 4. 사용자 공간으로의 복귀

#### a. 과정

- 이전에 저장한 사용자 프로그램의 context를 복원한다.

시스템 콜 인터페이스가 `sysret` 을 실행하여 return 값이 전달된다.

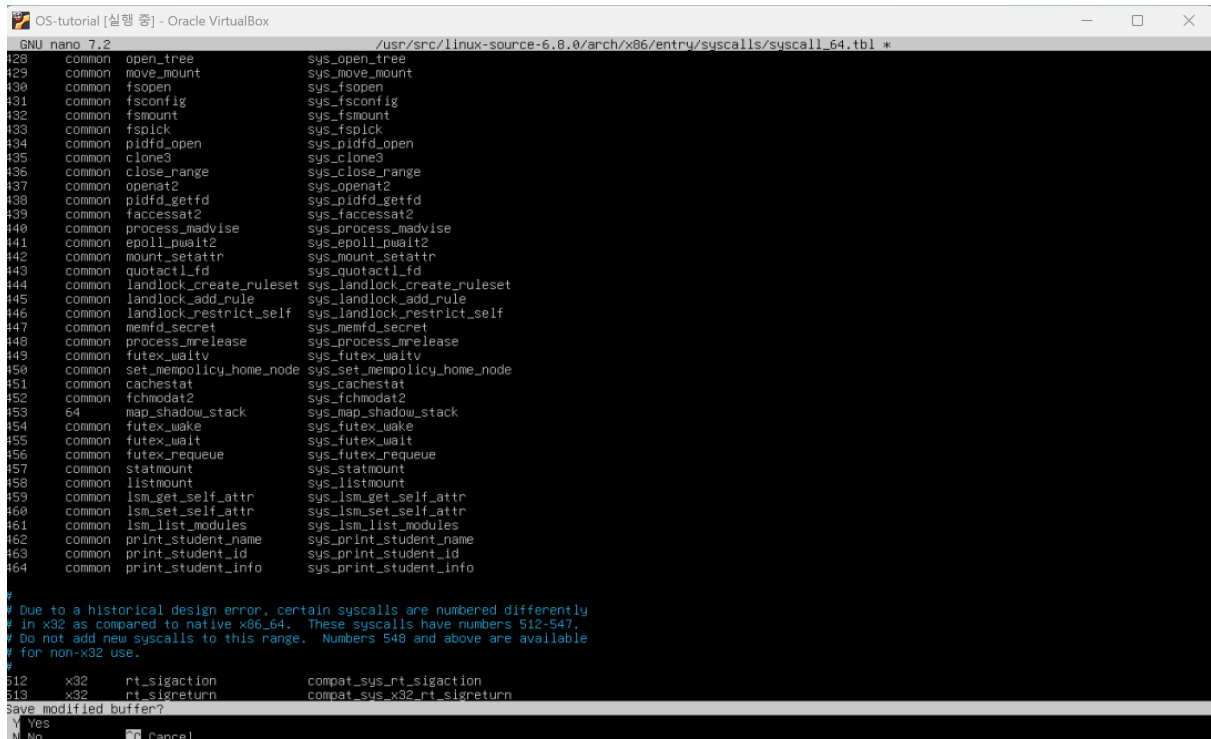
- CPU는 즉시 커널 모드에서 유저 모드로 전환하고, 사용자 프로그램은 `syscall` 을 호출했던 바로 다음 지점부터 실행을 재개한다.
- 래퍼 함수가 커널이 전달한 결과값을 확인하고 사용자에게 보여준다.

#### b. 목적

- 사용자는 커널 단에서 일어나는 복잡한 과정을 모르는 채로 편하게 `write` 시스템 콜을 사용할 수 있다.

## 시스템콜 생성 및 커널 컴파일 실습

### 1. 시스템콜 테이블( `syscall_64.tbl` )에 새로운 시스템콜을 등록

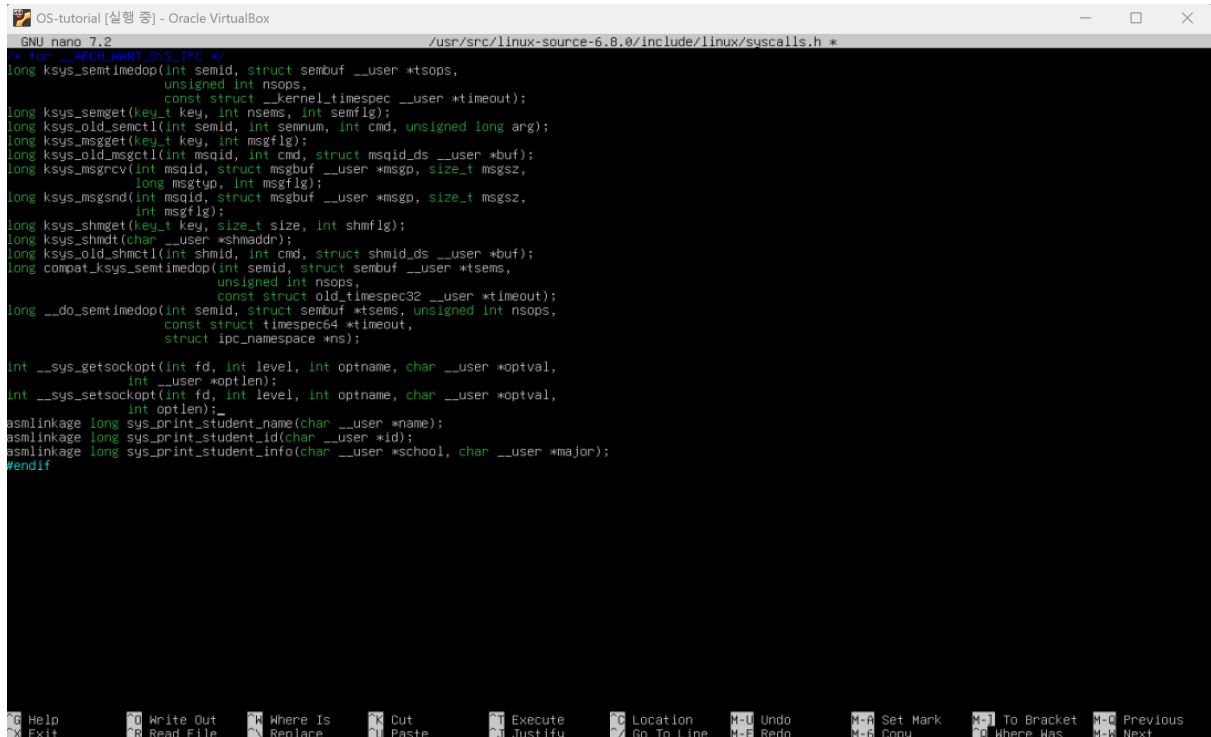


```
GNU nano 7.2 /usr/src/linux-source-6.8.0/arch/x86/entry/syscalls/syscall_64.tbl *
#
# Due to a historical design error, certain syscalls are numbered differently
# in x32 as compared to native x86_64. These syscalls have numbers 512-547.
# Do not add new syscalls to this range. Numbers 548 and above are available
# for non-x32 use.
#
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
#
612 x32 rt_sigaction compat_sys_rt_sigaction
613 x32 rt_sigreturn compat_sys_x32_rt_sigreturn
Save modified buffer?
[Y] Yes
[N] No [C] Cancel
```

- 462번 - print\_student\_name
- 463번 - print\_student\_id
- 464번 - print\_student\_info

- 커널은 사용자 프로그램으로부터 받은 시스템콜 식별 번호를 가지고, 시스템콜 테이블에서 매핑되는 함수를 찾아 실행한다. 따라서 직접 만든 함수를 시스템콜로 호출할 수 있게 하려면 이 테이블에 등록하는 과정이 필요하다.

### 2. `syscall.h` 에 새 시스템콜 등록



```
GNU nano 7.2 /usr/src/linux-source-6.8.0/include/linux/syscalls.h *
/* for __ARCH_WANT_SYS_IPC */
long ksys_semtimedop(int semid, struct sembuf __user *tsops,
                    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_msgget(key_t key, int msgflg);
long ksys_old_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_shmdt(char __user *shmaddr);
long ksys_old_shmctl(int shmid, int cmd, struct shmid_ds __user *buf);
long compat_ksys_semtimedop(int semid, struct sembuf __user *tsops,
                            unsigned int nsops,
                            const struct old_timespec32 __user *timeout);
long __do_semtimedop(int semid, struct sembuf *tsops, 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 long sys_print_student_name(char __user *name);
asmlinkage long sys_print_student_id(char __user *id);
asmlinkage long sys_print_student_info(char __user *school, char __user *major);
#endif
```

- `#endif` 위로 보이는 세 개의 프로토타입 함수(`sys_print_student_name`, `sys_print_student_id`, `sys_print_student_info`)를 추가했다.
- 컴파일하는 시점에 커널이 찾는 핸들러 함수들이 선언돼있어야 테이블에 올바르게 매핑할 수 있다.

### 3. `Makefile` 에 새 시스템콜 함수 파일 추가

- `new_syscall.o` 를 추가했다.
- 커널 빌드 대상이 되는 오브젝트 파일들의 목록에 `new_syscall.o` 를 추가함으로써, `new_syscall.c` 가 컴파일 대상에 포함되게 된다.

#### 4. `new_syscall.c` 작성

- `print_student_name`  
`char __user *name` : 사용자 공간에서 문자열 포인터 name을 인자로 받아 커널 버퍼 buf에 복사한다. 커널 로그에 "My Name is ..."을 출력한다.
- `print_student_id`  
`char __user *id` : 사용자 공간에서 문자열 포인터 id를 인자로 받아 커널 버퍼 buf에 복사한다. 커널 로그에 "My Student ID is ..."을 출력한다.

- `print_student_info`

`char __user *school`, `char __user *major`: 사용자 공간에서 문자열 포인터 `school`과 `major`를 받아 각각 커널 버퍼 `school_buf`, `major_buf`에 복사한다. 커널 로그에 "I go to ..."과 "I major in ..."을 출력한다.

## 5. `.config` 파일 수정

```

OS-tutorial [실형 중] - Oracle VirtualBox
GNU nano 7.2 .config *
CONFIG_CRYPTO_DEV_IAA_CRYPTO=m
# CONFIG_CRYPTO_DEV_IAA_CRYPTO_STATS is not set
CONFIG_CRYPTO_DEV_CHELSIO=m
CONFIG_CRYPTO_DEV_VIRTIO=m
CONFIG_CRYPTO_DEV_SAFEKCEL=m
CONFIG_CRYPTO_DEV_AWLOGIC_GXL=m
# CONFIG_CRYPTO_DEV_AWLOGIC_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

```

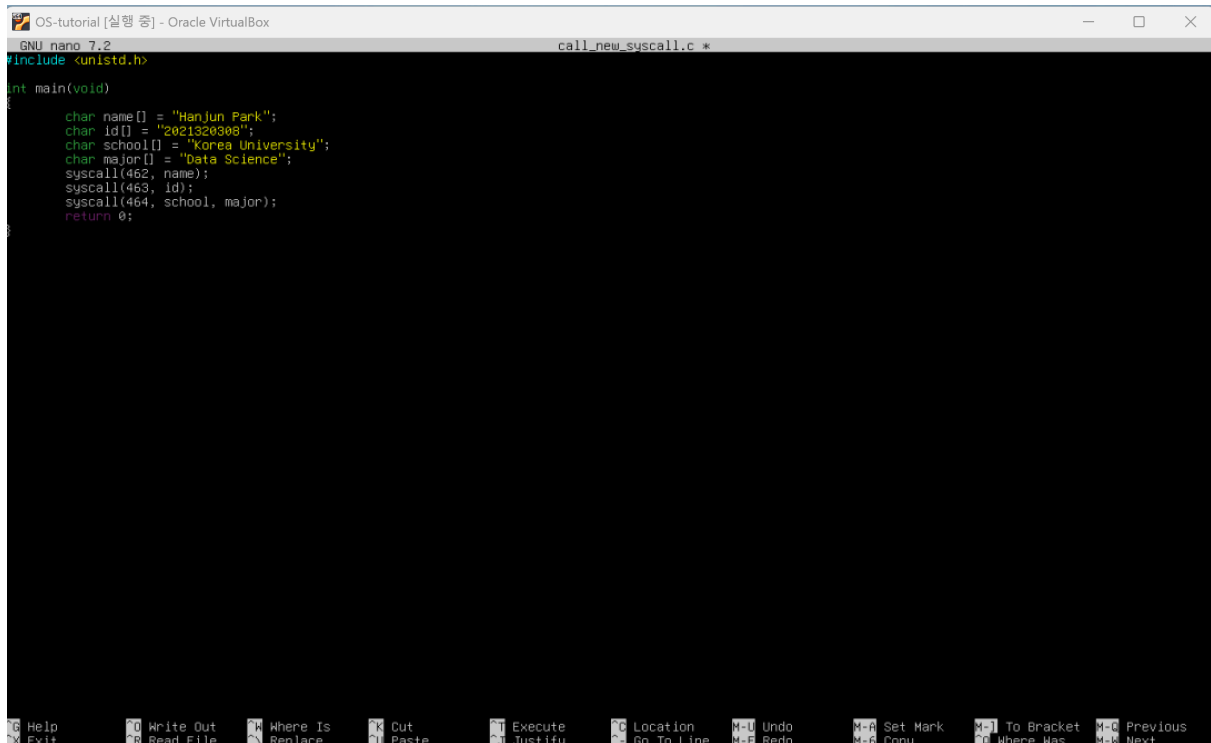
- 커널 빌드 및 부팅 시 인증 오류를 방지하기 위해 신뢰하는 키 목록과 거부된 키 목록을 비웠다. 즉, `CONFIG_SYSTEM_TRUSTED_KEYS = ""` 로 수정했고, `CONFIG_SYSTEM_REVOCATION_KEYS = ""`로 수정했다.

## 6. 커널 컴파일 후 커널 버전 확인

[illegible]

- 내가 직접 컴파일 한 커널 6.8.12 버전이 성공적으로 부팅된 모습을 의미한다. 즉, 앞서 정의한 시스템콜들이 성공적으로 컴파일되어 커널에 포함됐을 것이다.

## 7. `call_new_syscall.c` 작성



```
OS-tutorial [실행 중] - Oracle VirtualBox
GNU nano 7.2                                call_new_syscall.c *
#include <unistd.h>

int main(void)
{
    char name[] = "HanJun Park";
    char id[] = "2021320308";
    char school[] = "Korea University";
    char major[] = "Data Science";
    syscall(462, name);
    syscall(463, id);
    syscall(464, school, major);
    return 0;
}
```

- `syscall()` 함수를 직접 호출하여 번호와 인자를 명시적으로 전달했다. 각각 462번 시스템콜에 name이란 인자를, 463번 시스템콜에 id란 인자를, 464번 시스템콜에 school과 major란 인자를 전달했다.

## 8. `call_new_syscall.c` 컴파일 및 실행 후 `dmesg` 로 커널 메시지 출력

```
OS-tutorial [실행 중] - Oracle VirtualBox
[ 8.416850] vmwgfx 0000:00:02.0: [drm] fb0: vmwgfxdrmfb frame buffer device
[ 8.553169] workqueue: drm_fb_helper_damage_work hogged CPU for >13333us 4 times, consider switching to WQ_UNBOUND
[ 8.563782] EXT4-fs (sda2): mounted filesystem 0d689369-5e94-4a73-8663-4319516e764f r/w with ordered data mode. Quota mode: none.
[ 8.643101] audit: type=1400 audit(1744265470.072:2): apparmor="STATUS" operation="profile_load" profile="unconfined" name="Discord" pid=574 comm="apparmor_p
arser"
[ 8.644849] audit: type=1400 audit(1744265470.073:3): apparmor="STATUS" operation="profile_load" profile="unconfined" name="1password" pid=573 comm="apparmor
arser"
[ 8.644854] audit: type=1400 audit(1744265470.073:4): apparmor="STATUS" operation="profile_load" profile="unconfined" name="406F6E676F444220436F6070617373 pid
=575 comm="apparmor_parser"
[ 8.644856] audit: type=1400 audit(1744265470.073:5): apparmor="STATUS" operation="profile_load" profile="unconfined" name="QtWebEngineProcess" pid=576 comm=
"apparmor_parser"
[ 8.648106] audit: type=1400 audit(1744265470.077:6): apparmor="STATUS" operation="profile_load" profile="unconfined" name="buildah" pid=580 comm="apparmor_p
arser"
[ 8.648178] audit: type=1400 audit(1744265470.077:7): apparmor="STATUS" operation="profile_load" profile="unconfined" name="brave" pid=579 comm="apparmor_par
ser"
[ 8.649683] audit: type=1400 audit(1744265470.078:8): apparmor="STATUS" operation="profile_load" profile="unconfined" name="balena-etcher" pid=578 comm="appa
rmor_parser"
[ 8.650257] audit: type=1400 audit(1744265470.079:9): apparmor="STATUS" operation="profile_load" profile="unconfined" name="busybox" pid=581 comm="apparmor_p
arser"
[ 8.652226] audit: type=1400 audit(1744265470.081:10): apparmor="STATUS" operation="profile_load" profile="unconfined" name="cam" pid=582 comm="apparmor_pars
er"
[ 8.654016] audit: type=1400 audit(1744265470.081:11): apparmor="STATUS" operation="profile_load" profile="unconfined" name="ch-run" pid=584 comm="apparmor_p
arser"
[ 9.043327] workqueue: drm_fb_helper_damage_work hogged CPU for >13333us 8 times, consider switching to WQ_UNBOUND
[ 9.074688] cfg80211: Loading compiled-in X.509 certificates for regulatory database
[ 9.075029] Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[ 9.075148] Loaded X.509 cert 'wens: 61c08651aabd0cf94bd0ac7ff06c7248db18c600'
[ 9.106282] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[ 9.600750] NET: Registered PF_QIPCRTR protocol family
[ 9.678662] loop0: detected capacity change from 0 to 8
[ 10.107409] workqueue: drm_fb_helper_damage_work hogged CPU for >13333us 16 times, consider switching to WQ_UNBOUND
[ 19.289089] systemd-journald[329]: /var/log/journal/970a1b5543e84230bc706865e8f8a9df/user-1000.journal: Journal file uses a different sequence number ID, rot
ating.
[ 392.690549] My Name is Hanjun Park
[ 392.690577] My Student ID is 2021320308
[ 392.690599] I go to Korea University
[ 392.690561] I major in Data Science
ohj@2021320308:~/workspace$ sudo dmesg | tail
[ 9.106282] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[ 9.600750] NET: Registered PF_QIPCRTR protocol family
[ 9.678662] loop0: detected capacity change from 0 to 8
[ 10.107409] workqueue: drm_fb_helper_damage_work hogged CPU for >13333us 16 times, consider switching to WQ_UNBOUND
[ 19.289089] systemd-journald[329]: /var/log/journal/970a1b5543e84230bc706865e8f8a9df/user-1000.journal: Journal file uses a different sequence number ID, rot
ating.
[ 392.690549] My Name is Hanjun Park
[ 392.690577] My Student ID is 2021320308
[ 392.690599] I go to Korea University
[ 392.690561] I major in Data Science
[ 459.009817] workqueue: drm_fb_helper_damage_work hogged CPU for >13333us 32 times, consider switching to WQ_UNBOUND
ohj@2021320308:~/workspace$
```

- 앞서 `call_new_syscall.c` 의 인자들에 정의했던 `name("Hanjun Park")`, `id("2021320308")`, `school("Korea University")`, `major("Data Science")`가, 시스템콜 함수의 정의에 맞게 다 잘 출력되고 있다.