
Assignment #1. Syscall



과 목 명 : 운영체제

담 당 교 수 : 박태준 교수님

제 출 일 : 2025. 04. 15.

학 과 : 지역·바이오시스템공학과

학 번 : 184128

이 름 : 박 지 환

#01. Compiling Linux Kernel

기대 결과물) 리눅스 커널을 컴파일 하는 과정과 설명이 담긴 레포트

Step 1) 가상머신 또는 남는 PC에 Ubuntu 24.04.2 LTS를 설치하세요.

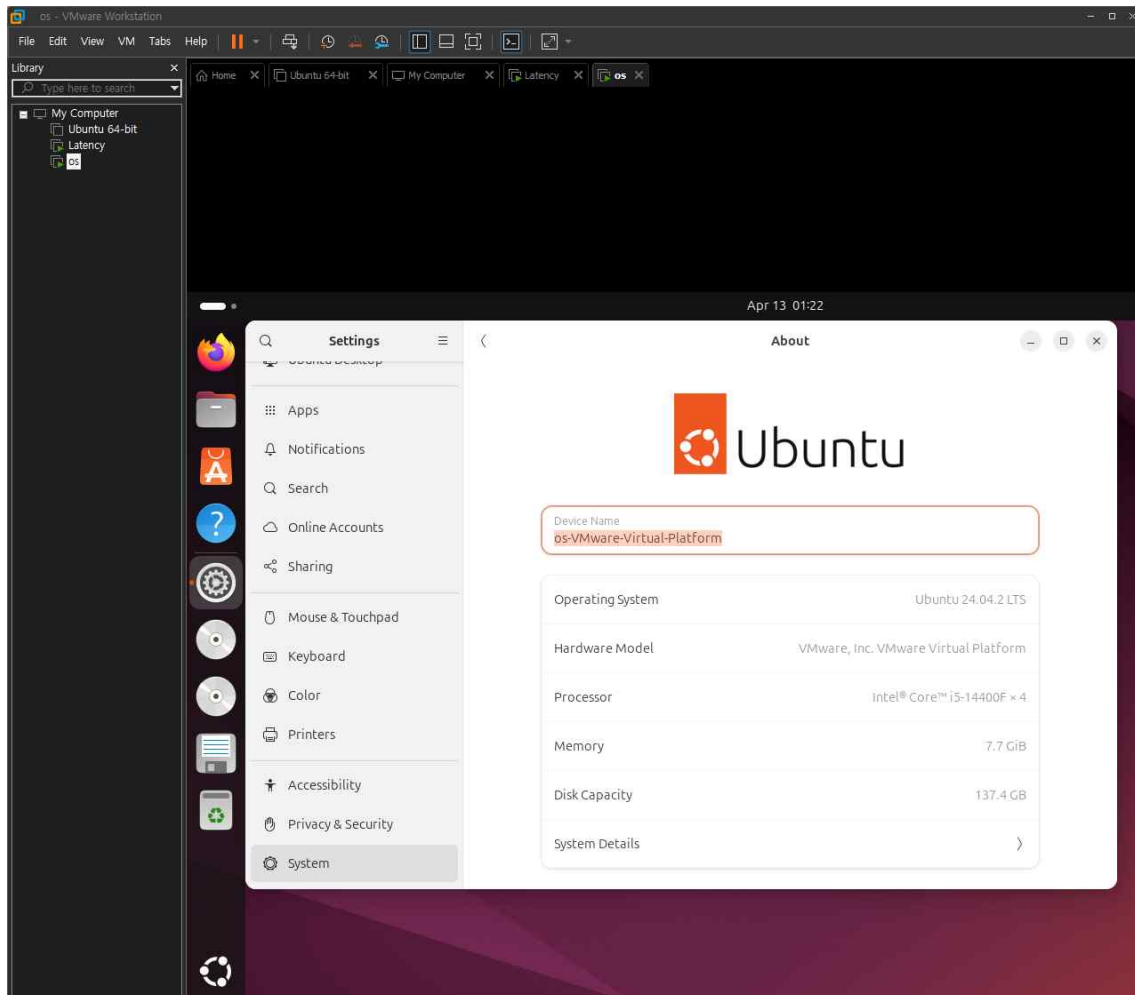


그림 1. 본 과제를 위해 VMware Workstation Pro 가상머신에 Ubuntu 24.04.2 LTS (AMD64)를 설치하였습니다.

- CPU: 4코어
- 메모리: 8GB
- 디스크 용량: 128GB

Step 2) 커널 소스코드를 다운받고, 커널을 빌드 후 여러분의 환경에 적용시켜보세
요.

```
os@os-VMware-Virtual-Platform: $ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.13.11.tar.xz
--2025-04-11 20:52:36-- https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.13.11.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 146.75.49.176, 2a04:4e42:7c::432
Connecting to cdn.kernel.org (cdn.kernel.org)|146.75.49.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 148610520 (142M) [application/x-xz]
Saving to: 'linux-6.13.11.tar.xz'

linux-6.13.11.tar.xz      100%[=====] 141.73M  11.3MB/s   in 13s
2025-04-11 20:52:50 (11.2 MB/s) - 'linux-6.13.11.tar.xz' saved [148610520/148610520]

os@os-VMware-Virtual-Platform: $ tar -xvf linux-6.13.11.tar.xz
```

그림 2. 리눅스 커널 소스 버전 6.13.11을 다운로드 및 압축 해제하여 컴파일을 진행하였습니다.

- 교수님께서 과제 참고자료로 올려주신 아래 유튜브 링크를 참고하여 6.13 커널을 설치하였습니다.

→ <https://www.youtube.com/watch?v=i6-uT5yJg7o>

```
os@os-VMware-Virtual-Platform:~$ sudo add-apt-repository ppa:cappelikan/ppa
Repository: 'Types: deb
URIs: https://ppa.launchpadcontent.net/cappelikan/ppa/ubuntu/
Suites: noble
Components: main
'
Description:
Mainline Ubuntu Kernel Installer https://github.com/bkw777/mainline
More info: https://launchpad.net/~cappelikan/+archive/ubuntu/ppa
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://kr.archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://kr.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:4 http://kr.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:5 https://ppa.launchpadcontent.net/cappelikan/ppa/ubuntu noble InRelease [17.8 kB]
Get:6 https://ppa.launchpadcontent.net/cappelikan/ppa/ubuntu noble/main amd64 Packages [588 B]
Get:7 https://ppa.launchpadcontent.net/cappelikan/ppa/ubuntu noble/main Translation-en [316 B]
Fetched 145 kB in 10s (14.4 kB/s)
Reading package lists... Done
```

그림 3. Install Kernel 6.13 using the Mainline Kernel GUI tool

→ sudo add-apt-repository ppa:cappelikan/ppa

```
os@os-VMware-Virtual-Platform:~$ sudo apt update
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 https://ppa.launchpadcontent.net/cappelikan/ppa/ubuntu noble InRelease
Hit:3 http://kr.archive.ubuntu.com/ubuntu noble InRelease
Hit:4 http://kr.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:5 http://kr.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
92 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

그림 4. Install Kernel 6.13 using the Mainline Kernel GUI tool

→ sudo apt update

```

os@os-VMware-Virtual-Platform:~$ sudo apt install mainline
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  aria2 libaria2-0 libssh2-1t64
The following NEW packages will be installed:
  aria2 libaria2-0 libssh2-1t64 mainline
0 upgraded, 4 newly installed, 0 to remove and 92 not upgraded.
Need to get 1,794 kB of archives.
After this operation, 6,794 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://kr.archive.ubuntu.com/ubuntu noble/main amd64 libssh2-1t64 amd64 1.11.0-4.1build2 [120 kB]
Get:2 http://kr.archive.ubuntu.com/ubuntu noble/universe amd64 libaria2-0 amd64 1.37.0+debian-1build3 [1,105 kB]
Get:3 http://kr.archive.ubuntu.com/ubuntu noble/universe amd64 aria2 amd64 1.37.0+debian-1build3 [387 kB]
Get:4 https://ppa.launchpadcontent.net/cappelikan/ppa/ubuntu noble/main amd64 mainline amd64 1.4.13-0~202504060550~ubuntu24.04.1 [182 kB]
Fetched 1,794 kB in 6s (297 kB/s)
Selecting previously unselected package libssh2-1t64:amd64.
(Reading database ... 152855 files and directories currently installed.)
Preparing to unpack .../libssh2-1t64_1.11.0-4.1build2_amd64.deb ...
Unpacking libssh2-1t64:amd64 (1.11.0-4.1build2) ...
Selecting previously unselected package libaria2-0:amd64.
Preparing to unpack .../libaria2-0_1.37.0+debian-1build3_amd64.deb ...
Unpacking libaria2-0:amd64 (1.37.0+debian-1build3) ...
Selecting previously unselected package aria2.
Preparing to unpack .../aria2_1.37.0+debian-1build3_amd64.deb ...
Unpacking aria2 (1.37.0+debian-1build3) ...
Selecting previously unselected package mainline.
Preparing to unpack .../mainline_1.4.13-0~202504060550~ubuntu24.04.1_amd64.deb ...
Unpacking mainline (1.4.13-0~202504060550~ubuntu24.04.1) ...
Setting up libssh2-1t64:amd64 (1.11.0-4.1build2) ...
Setting up libaria2-0:amd64 (1.37.0+debian-1build3) ...
Setting up aria2 (1.37.0+debian-1build3) ...
Setting up mainline (1.4.13-0~202504060550~ubuntu24.04.1) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu3) ...
Processing triggers for libc-bin (2.39-0ubuntu8.4) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for desktop-file-utils (0.27-2build1) ...

```

그림 5. Install Kernel 6.13 using the Mainline Kernel GUI tool

→ sudo apt install mainline

Step 3) ‘uname -a’ 및 ‘cat /etc/os-release’ 명령어를 통해 커널이 성공적으로 컴파일 되었음을 확인해보세요.

```

os@os-VMware-Virtual-Platform:~$ uname -r
6.11.0-21-generic
os@os-VMware-Virtual-Platform:~$ uname -a
Linux os-VMware-Virtual-Platform 6.11.0-21-generic #21-24.04.1-Ubuntu SMP PREEMPT_DYNAMIC Mon Feb 24 16:52:15 UTC 2 x86_64 x
86_64 x86_64 GNU/Linux
os@os-VMware-Virtual-Platform:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04.2 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.2 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo

```

그림 6. 시스템 재부팅 후 명령어를 통해 커널이 정상적으로 적용되었는지 확인하였습니다.

- 위 결과를 통해 6.13.11 커널이 성공적으로 적용되었음을 확인할 수 있었습니다.

#02. Adding My First System Calls

기대 결과물) syscall을 추가하는 방법과 과정이 담긴 레포트, 그리고 도중에 생성된 소스코드 파일 및 기타 추가적인 파일들 (e.g., Makefile 등)

- ~/linux-6.13.11/kernel/sys_hello.c
- ~/linux-6.13.11/kernel/Makefile
- ~/linux-6.13.11/arch/x86/entry/syscalls/syscall_64.tbl
- ~/linux-6.13.11/include/linux/syscalls.h
- ~/linux-6.13.11/.config
- ~/syscall_test.c

참고 자료

1. https://junshim.github.io/linux%20kernel%20study/Linux_Kernel_Compile/
2. https://junshim.github.io/linux%20kernel%20study/Add_a_New_System_Call/

Step 1) 여러분 환경에 위에서 제시된 'sys_hello' 시스템콜을 추가하세요. 새로이 커널 컴파일이 필요할겁니다.

```
os@os-VMware-Virtual-Platform:~/linux-6.13.11$ ls
arch  COPYING  Documentation  include  ipc  kernel  MAINTAINERS  net  samples  sound  virt
block  CREDITS  drivers  init  Kbuild  lib  Makefile  README  scripts  tools  vmlinux-gdb.py
certs  crypto  fs  io_uring  Kconfig  LICENSES  mm  rust  security  usr

os@os-VMware-Virtual-Platform:~/linux-6.13.11$ cd kernel/
os@os-VMware-Virtual-Platform:~/linux-6.13.11/kernel$ ls
acct.c          cred.c          kallsyms_internal.h  module          scftorture.c      tsacct.c
acct.o          cred.o          kallsyms.o           module_signature.c  sched             tsacct.o
async.c         debug           kallsyms_selftest.c  module_signature.o  scs.c            ucount.c
async.o         delayacct.c     kallsyms_selftest.h  modules.order      seccomp.c         ucount.o
audit.c         delayacct.o     kcmp.c               notifier.c         seccomp.o         uid16.c
auditfilter.c   dma             kcmp.o               notifier.o         signal.c          uid16.h
auditfilter.o   dma.c           Kconfig.freezer      nsproxy.c         signal.o          uid16.o
audit_fsnotify.c  dma.o          Kconfig.hz           nsproxy.o         snpboot.c         umh.c
audit_fsnotify.o elfcorehdr.c    Kconfig.kexec        padata.c          snpboot.h         umh.o
audit.h          elfcorehdr.o   Kconfig.locks        padata.o          snpboot.o         up.c
audit.o          entry           Kconfig.preempt      panic.c           smp.c            user.c
auditsc.c       events          kcov.c               panic.o           smp.o            usermode_driver.c
auditsc.o       exec_domain.c  kcsan                params.c          softirq.c         user_namespace.c
audit_tree.c    exec_domain.o  kexec.c              params.o          softirq.o         user_namespace.o
audit_tree.o    exit.c         kexec_core.c         pid.c             stackleak.c       user.o
audit_watch.c   exit.h         kexec_core.o         pid_namespace.c   stacktrace.c      user-return-notifier.c
audit_watch.o   exit.o         kexec_elf.c          pid_namespace.o   stacktrace.o      user-return-notifier.o
backtracetest.c extable.c      kexec_file.c         pid.o             static_call.c      utsname.c
bounds.c        extable.o      kexec_file.o         pid_sysctl.h      static_call_inline.c  utsname.o
bounds.s        fail_function.c kexec_internal.h     power            static_call_inline.o  utsname_sysctl.c
bpf             fork.c         kexec.o              printk            static_call.o        utsname_sysctl.o
built-in.a      fork.o         kheaders.c           profile.c         stop_machine.c     vhost_task.c
capability.c    freezer.c      kheaders_data.tar.xz  profile.o         stop_machine.o     vhost_task.o
capability.o     freezer.o     kheaders.md5         ptrace.c          sys.c              vmcore_info.c
cfi.c           futex          kheaders.mod         ptrace.o          sysctl.c            vmcore_info.o
cggroup         gcov           kheaders.o           range.c           sysctl.o            watchdog_buddy.c
compat.c        gen_kheaders.sh kprobes.c            range.o           sysctl-test.c      watchdog.c
compat.o        groups.c       kprobes.o            rcu               sys_ni.c            watchdog.o
configs         groups.o       ksyms_common.c       reboot.c          sys_ni.o            watchdog_perf.c
configs.c       hung_task.c   ksyms_common.o       reboot.o          sys.o               watchdog_perf.o
context_tracking.c hung_task.o   ksysfs.c             regset.c          taskstats.c        watch_queue.c
context_tracking.o iomem.c      ksysfs.o             regset.o          taskstats.o        watch_queue.o
cpu.c           iomem.o       kthread.c            relay.c           task_work.c        workqueue.c
cpu.o           irq            kthread.o            relay.o           task_work.o        workqueue_internal.h
cpu_pm.c        irq_work.c    latencytop.c         resource.c         time               workqueue.o
crash_core.c    irq_work.o    latencytop.o         resource_kunit.c  torture.c          trace
crash_core.o    jump_label.c  livepatch            resource.o         tracepoint.c       trace
crash_reserve.c jump_label.o   locking              rseq.c            tracepoint.o
crash_reserve.o kallsyms.c    Makefile             rseq.o            tracepoint.o

os@os-VMware-Virtual-Platform:~/linux-6.13.11/kernel$ touch sys_hello.c
os@os-VMware-Virtual-Platform:~/linux-6.13.11/kernel$ nano sys_hello.c
```

그림 7. 커널 디렉토리로 이동하여 sys_hello.c 파일을 생성하고, nano 편집기를 열어 해당 파일에 그림 8과 같이 코드를 작성하였습니다.

Assignment #1. Syscall

```
GNU nano 7.2 sys_hello.c *
#include <linux/kernel.h>
#include <linux/syscalls.h>

asmlinkage long __sys_hello(void) {
    printk("Hello, Jihwan Park!\n");
    printk("184128\n");
    return 0;
}
```

그림 8. 해당 시스템 콜이 호출되면 커널 메시지에 인사, 이름 그리고 학번을 호출하고 시스템 호출이 성공적으로 실행되었음을 나타내는 0을 반환하는 코드를 작성하였습니다.

→ ~/linux-6.13.11/kernel/sys_hello.c

```
GNU nano 7.2 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 ksyzfs.o cred.o reboot.o \
        async.o range.o smptboot.o ucount.o regset.o ksyms_common.o sys_hello.o
```

그림 9. 위에서 구현한 새로운 시스템 콜인 sys_hello.c를 Makefile에 등록하여 다른 시스템 콜들과 함께 컴파일될 수 있도록 해주었습니다.

→ ~/linux-6.13.11/kernel/Makefile

- 구현한 시스템 콜이 sys_hello.c이기 때문에 sys_hello.o라고 추가하였습니다.

```
GNU nano 7.2 syscall_64.tbl *
517 x32 recvfrom compat_sys_recvfrom
518 x32 sendmsg compat_sys_sendmsg
519 x32 recvmsg compat_sys_recvmsg
520 x32 execve compat_sys_execve
521 x32 ptrace compat_sys_ptrace
522 x32 rt_sigpending compat_sys_rt_sigpending
523 x32 rt_sigtimedwait compat_sys_rt_sigtimedwait_time64
524 x32 rt_sigqueueinfo compat_sys_rt_sigqueueinfo
525 x32 sigaltstack compat_sys_sigaltstack
526 x32 timer_create compat_sys_timer_create
527 x32 mq_notify compat_sys_mq_notify
528 x32 kexec_load compat_sys_kexec_load
529 x32 waitid compat_sys_waitid
530 x32 set_robust_list compat_sys_set_robust_list
531 x32 get_robust_list compat_sys_get_robust_list
532 x32 vmsplce sys_vmsplce
533 x32 move_pages sys_move_pages
534 x32 preadv compat_sys_preadv64
535 x32 pwritev compat_sys_pwritev64
536 x32 rt_tgsigqueueinfo compat_sys_rt_tgsigqueueinfo
537 x32 recvmmsg compat_sys_recvmmsg_time64
538 x32 sendmmsg compat_sys_sendmmsg
539 x32 process_vm_readv sys_process_vm_readv
540 x32 process_vm_writev sys_process_vm_writev
541 x32 setsockopt sys_setsockopt
542 x32 getsockopt sys_getsockopt
543 x32 io_setup compat_sys_io_setup
544 x32 io_submit compat_sys_io_submit
545 x32 execveat compat_sys_execveat
546 x32 preadv2 compat_sys_preadv64v2
547 x32 pwritev2 compat_sys_pwritev64v2
548 common sys_hello __sys_hello
# This is the end of the legacy x32 range. Numbers 548 and above are
# not special and are not to be used for x32-specific syscalls.
```

그림 10. 구현한 시스템 콜을 시스템 콜 테이블에 등록해주기 위해 548번 인덱스에 추가하였습니다.

→ ~/linux-6.13.11/arch/x86/entry/syscalls/syscall_64.tbl

```

GNU nano 7.2 syscalls.h *
long ksys_old_sendctl(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 *tsems,
                            unsigned int nsops,
                            const struct old_timespec32 __user *timeout);
long __do_semtimedop(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 long __sys_hello(void);
#endif

```

그림 11. 구현한 시스템 콜을 시스템 콜 헤더파일에 등록하였습니다.

→ ~/linux-6.13.11/include/linux/syscalls.h

- 이후 커널 컴파일 및 재부팅을 위해 커널 의존성 패키지 설치 및 Configuration 파일을 생성합니다.

```
os@os-virtual-machine:~/linux-6.13.11$ sudo apt install build-essential libncurses-dev bison flex libssl-dev libelf-dev
```

그림 12. 의존성 패키지 설치

```

os@os-virtual-machine:~/linux-6.13.11$ cp /boot/config-$(uname -r) .config
os@os-virtual-machine:~/linux-6.13.11$ make olddefconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX scripts/kconfig/lexer.lex.c
YACC scripts/kconfig/parser.tab.[ch]
HOSTCC scripts/kconfig/lexer.lex.o
HOSTCC scripts/kconfig/menu.o
HOSTCC scripts/kconfig/parser.tab.o
HOSTCC scripts/kconfig/preprocess.o
HOSTCC scripts/kconfig/symbol.o
HOSTCC scripts/kconfig/util.o
HOSTLD scripts/kconfig/conf
.config:7436:warning: symbol value 'm' invalid for FB_BACKLIGHT
.config:10992:warning: symbol value 'm' invalid for ANDROID_BINDER_IPC
.config:10993:warning: symbol value 'm' invalid for ANDROID_BINDERFS
#
# configuration written to .config
#

```

그림 13. 현재 시스템의 설정 기반으로 커널 Configuration 파일을 생성하였습니다.

```
os@os-virtual-machine:~/linux-6.13.11$ make -j$(nproc)
```

그림 14. CPU의 코어 수만큼 커널을 빌드하는 명령어를 실행하였습니다.


```
os@os-virtual-machine: ~/linux-6.13.11$ make -j$(nproc)
SYSHDR arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_x32.h
SYSHDR arch/x86/include/generated/asm/unistd_64_x32.h
SYSTBL arch/x86/include/generated/asm/syscalls_64.h
DESCEND objtool
INSTALL libsubcmd_headers
CC arch/x86/kernel/asm-offsets.s
CALL scripts/checksyscalls.sh
make[3]: *** No rule to make target 'debian/canonical-certs.pem', needed by 'certs/x509_certificate_list'. Stop.
make[2]: *** [scripts/Makefile.build:442: certs] Error 2
make[2]: *** Waiting for unfinished jobs...
```

그림 15. 에러 발생!

- 참고자료: <https://coding-babo.tistory.com/41>

리눅스 컴파일을 하던 도중 밑과 같은 에러가 발생했습니다.

```
No rule to make target 'debian/canonical-certs.pem', needed by 'certs/x509_certificate_list'.
```

이를 해결해주기 위해서는

```
CONFIG_SYSTEM_TRUSTED_KEYS="debian/canonical-certs.pem" ->
```

```
CONFIG_SYSTEM_TRUSTED_KEYS = ""
```

밑의 문장과 같이 변경해줘야 한다고 합니다.

이를 해주기 위해 쉘 창에 밑과 같은 명령어를 쳐주면 됩니다.

```
scripts/config --disable SYSTEM_TRUSTED_KEYS
```

++

revocation 키에 관련된 에러도 나오는 것 같길래

```
scripts/config --disable SYSTEM_REVOCATION_KEYS
```

위 명령어를 넣어줬더니 다음단계로 진행이 되었습니다.

```

GNU nano 7.2                                     .config
# CONFIG_CRYPTO_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="debian/canonical-certs.pem"
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="debian/canonical-revoked-certs.pem"
# CONFIG_SYSTEM_BLACKLIST_AUTH_UPDATE is not set
# end of Certificates for signature checking

```

그림 16. 위의 에러를 해결하기 위해 CONFIG_SYSTEM_TRUSTED_KEYS 란을 공백으로 설정 하였습니다.

→ ~/linux-6.13.11/.config → **CONFIG_SYSTEM_TRUSTED_KEYS=""**

- 이후 make olddefconfig를 통해 변경 사항을 적용하고 다시 빌드 컴파일을 진행하였습니다.

```

os@os-virtual-machine:~/linux-6.13.11$ make -j$(nproc)
SYNC include/config/auto.conf.cmd
DESCEND objtool
INSTALL libsubcmd headers
CALL scripts/checksyscalls.sh
CC io_uring/io_uring.o
AS arch/x86/lib/clear_page_64.o
HOSTCC certs/extract-cert
CC lib/math/div64.o
CC lib/math/gcd.o
CC lib/math/lcm.o
COPY certs/x509.genkey
CC certs/blacklist.o
CC arch/x86/lib/cmdline.o
CC lib/math/int_log.o
CC lib/math/int_pow.o
AS arch/x86/lib/cmpxchg16b_emu.o
CC arch/x86/lib/copy_mc.o
CC lib/math/int_sqrt.o
CC lib/math/reciprocal_div.o
CC lib/math/rational.o
AS arch/x86/lib/copy_mc_64.o
AS arch/x86/lib/copy_page_64.o
make[3]: *** No rule to make target 'debian/canonical-revoked-certs.pem', needed by 'certs/x509_revocation_list'. Stop.
make[3]: *** Waiting for unfinished jobs....
GEN certs/blacklist_hash_list
make[2]: *** [scripts/Makefile.build:442: certs] Error 2
make[2]: *** Waiting for unfinished jobs....

```

그림 17. 또 다른 에러 발생..

```

GNU nano 7.2 .config
# 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="debian/canonical-revoked-certs.pem"
# CONFIG_SYSTEM_BLACKLIST_AUTH_UPDATE is not set
# end of Certificates for signature checking

```

그림 18. 위의 에러를 해결하기 위해 CONFIG_SYSTEM_REVOCATION_KEYS 란을 공백으로 설정하였습니다.

→ ~/linux-6.13.11/.config → **CONFIG_SYSTEM_REVOCATION_KEYS=""**

```
os@os-virtual-machine:~/linux-6.13.11$ sudo make modules_install
```

그림 19. 다음 코드를 통해 커널 빌드 후 모듈을 설치하였습니다.

```

os@os-virtual-machine:~/linux-6.13.11$ sudo make install
INSTALL /boot

*** Missing file: arch/x86/boot/bzImage
*** You need to run "make" before "make install".

make[1]: *** [arch/x86/Makefile:321: install] Error 1
make: *** [Makefile:251: _sub-make] Error 2

```

그림 20. make 명령을 실행했지만 실제로 bzImage가 생성되지 않았다는 에러가 발생했고, 커널 이미지만 다시 빌드해보기 위해 아래와 같은 코드를 실행하였습니다.

→ make bzImage -j\$(nproc)

```

os@os-virtual-machine:~/linux-6.13.11$ make bzImage -j$(nproc)
DESCEND objtool
INSTALL libsubcmd_headers
CALL scripts/checksyscalls.sh
MODPOST vmlinux.symvers
UPD include/generated/utsversion.h
CC init/version-timestamp.o
KSYMS .tmp_vmlinux0.kallsyms.S
AS .tmp_vmlinux0.kallsyms.o
LD .tmp_vmlinux1
ld: arch/x86/entry/syscall_64.o: in function 'x64_sys_call':
/home/os/linux-6.13.11/.arch/x86/include/generated/asm/syscalls_64.h:549:(.text+0x16ee): undefined reference to '__x64__sys_hello'
ld: arch/x86/entry/syscall_64.o:(.rodata+0x1120): undefined reference to '__x64__sys_hello'
make[2]: *** [scripts/Makefile.vmlinux:77: vmlinux] Error 1
make[1]: *** [/home/os/linux-6.13.11/Makefile:1230: vmlinux] Error 2
make: *** [Makefile:251: _sub-make] Error 2

```

그림 21. 커널에 새 시스템 콜 sys_hello를 추가하면서 다음과 같은 연결 오류가 발생했습니다. 즉, 커널이 __x64__sys_hello라는 시스템 콜을 찾지 못해 빌드가 실패한 것이었습니다.

```

GNU nano 7.2 sys_hello.c *
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(hello) {
    printk(KERN_INFO "Hello, Jihwan Park!\n");
    printk(KERN_INFO "184128\n");
    return 0;
}

```

그림 22. 에러 해결을 위해 기존의 sys_hello.c에서 SYSCALL_DEFINE0(hello)로 정의하여 커널을 매핑했습니다.

```

GNU nano 7.2 Makefile *
obj-$(CONFIG_MIS_IOMEM) += iomem.o
obj-$(CONFIG_MSEQ) += rseq.o
obj-$(CONFIG_WATCH_QUEUE) += watch_queue.o

obj-$(CONFIG_RESOURCE_KUNIT_TEST) += resource_kunit.o
obj-$(CONFIG_SYSCALL_KUNIT_TEST) += syscall-test.o

CFLAGS_stackleak.o += $(DISABLE_STACKLEAK_PLUGIN)
obj-$(CONFIG_GCC_PLUGIN_STACKLEAK) += stackleak.o
KASAN_SANITIZE_stackleak.o := n
KCSAN_SANITIZE_stackleak.o := n
KCOV_INSTRUMENT_stackleak.o := n

obj-$(CONFIG_SCF_TORTURE_TEST) += scftorture.o

$(obj)/configs.o: $(obj)/config_data.gz
targets += config_data config_data.gz
$(obj)/config_data.gz: $(obj)/config_data FORCE
    $(call if_changed,gzip)

filechk_cat = cat <

$(obj)/config_data: $(KCONFIG_CONFIG) FORCE
    $(call filechk,cat)

$(obj)/kheaders.o: $(obj)/kheaders_data.tar.xz
quiet_cmd_genikh = CHK      $(obj)/kheaders_data.tar.xz
cmd_genikh = $(CONFIG_SHELL) $(srcfile)/kernel/gen_kheaders.sh $@
$(obj)/kheaders_data.tar.xz: FORCE
    $(call cmd,genikh)

clean-files += kheaders_data.tar.xz kheaders.md5
obj-y += sys_hello.o

```

그림 23. 커널 Makefile에 sys_hello.o를 다음과 같이 수정하였습니다.

```

GNU nano 7.2 syscall_64.tbl *
517 x32 recvfrom compat_sys_recvfrom
518 x32 sendmsg compat_sys_sendmsg
519 x32 recvmsg compat_sys_recvmsg
520 x32 execve compat_sys_execve
521 x32 ptrace compat_sys_ptrace
522 x32 rt_sigpending compat_sys_rt_sigpending
523 x32 rt_sigtimedwait compat_sys_rt_sigtimedwait_time64
524 x32 rt_sigqueueinfo compat_sys_rt_sigqueueinfo
525 x32 sigaltstack compat_sys_sigaltstack
526 x32 timer_create compat_sys_timer_create
527 x32 mq_notify compat_sys_mq_notify
528 x32 kexec_load compat_sys_kexec_load
529 x32 waitid compat_sys_waitid
530 x32 set_robust_list compat_sys_set_robust_list
531 x32 get_robust_list compat_sys_get_robust_list
532 x32 vmsplice sys_vmsplice
533 x32 move_pages sys_move_pages
534 x32 preadv compat_sys_preadv64
535 x32 pwritev compat_sys_pwritev64
536 x32 rt_tgsigqueueinfo compat_sys_rt_tgsigqueueinfo
537 x32 recvmsg compat_sys_recvmsg_time64
538 x32 sendmsg compat_sys_sendmsg
539 x32 process_vm_readv sys_process_vm_readv
540 x32 process_vm_writev sys_process_vm_writev
541 x32 setsockopt sys_setsockopt
542 x32 getsockopt sys_getsockopt
543 x32 io_setup compat_sys_io_setup
544 x32 io_submit compat_sys_io_submit
545 x32 execveat compat_sys_execveat
546 x32 preadv2 compat_sys_preadv64v2
547 x32 pwritev2 compat_sys_pwritev64v2
548 common hello sys_hello
# This is the end of the legacy x32 range. Numbers 548 and above are
# not special and are not to be used for x32-specific syscalls.

```

그림 24. 그리고 기존에 사용했던 __sys_hello의 앞에 작성한 언더바에서 에러가 발생하는 것 같이 수정하였습니다.

```

os@os-virtual-machine:~/linux-6.13.11$ make clean
CLEAN arch/x86/entry/vdso
CLEAN arch/x86/kernel/cpu
CLEAN arch/x86/kernel
CLEAN arch/x86/purgatory
CLEAN arch/x86/realmode/rm
CLEAN arch/x86/tools
CLEAN arch/x86/lib
CLEAN certs
CLEAN drivers/firmware/efi/libstub
CLEAN drivers/scsi
CLEAN drivers/tty/vt
CLEAN fs/unicode
CLEAN init
CLEAN kernel/debug/kdb
CLEAN lib/test_fortify
CLEAN lib
CLEAN security/apparmor
CLEAN security/selinux
CLEAN security/tomoyo
CLEAN usr
CLEAN .
CLEAN vmlinux.symvers modules.builtin modules.builtin.modinfo vmlinux.objs vmlinux.export.c
os@os-virtual-machine:~/linux-6.13.11$ make bzImage -j$(nproc)

```

그림 25. 이후 빌드를 다시 시도하였습니다.

```

Kernel: arch/x86/boot/bzImage is ready (#4)
os@os-virtual-machine:~/linux-6.13.11$

```

그림 26. 커널 이미지 빌드 성공!

```

os@os-virtual-machine:~/linux-6.13.11$ sudo make modules_install
[sudo] password for os:
make[2]: *** No rule to make target 'modules.order', needed by '/lib/modules/6.13.11/modules.order'. Stop.
make[1]: *** [/home/os/linux-6.13.11/Makefile:1895: modules_install] Error 2
make: *** [Makefile:251: _sub-make] Error 2
os@os-virtual-machine:~/linux-6.13.11$ make modules -j$(nproc)

```

그림 27. 빌드 후 모듈을 설치하려는 과정에서 모듈 빌드가 되어있지 않다는 에러를 발견하고 커널에서 설정된 모듈들을 먼저 빌드하였습니다. 이후 다시 `sudo make modules_install`을 통해 모듈을 설치하였습니다.

```

os@os-virtual-machine:~/linux-6.13.11$ sudo make install
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.13.11 /boot/vmlinuz-6.13.11
update-initramfs: Generating /boot/initrd.img-6.13.11
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 6.13.11 /boot/vmlinuz-6.13.11
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.13.11 /boot/vmlinuz-6.13.11
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 6.13.11 /boot/vmlinuz-6.13.11
I: /boot/initrd.img is now a symlink to initrd.img-6.13.11
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.13.11 /boot/vmlinuz-6.13.11
Sourcing file '/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.13.11
Found initrd image: /boot/initrd.img-6.13.11
Found linux image: /boot/vmlinuz-6.11.0-21-generic
Found initrd image: /boot/initrd.img-6.11.0-21-generic
Found memtest86+x64 image: /boot/memtest86+x64.bin
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done

```

그림 28. 이후 커널 이미지를 설치하고 재부팅하였습니다.

Step 2) 잘 동작하는지 동작 확인!

```

GNU nano 7.2                                syscall_test.c *
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/kernel.h>

#define sys_hello 548

int main(int argc, char *argv[]) {
    int ret = syscall(sys_hello);

    return 0;
}

```

그림 29. 시스템 콜이 잘 만들어졌는지 테스트하기 위해 위와 같은 코드를 작성하였습니다.

```

os@os-virtual-machine:~$ sudo dmesg | grep "Hello, Jihwan Park!"
[ 578.168255] Hello, Jihwan Park!
os@os-virtual-machine:~$ sudo dmesg | grep "184128"
[ 578.168259] 184128

```

그림 30. 시스템 콜 테스트 성공!

```

os@os-virtual-machine:~$ sudo update-grub
Sourcing file '/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.13.11
Found initrd image: /boot/initrd.img-6.13.11
Found linux image: /boot/vmlinuz-6.11.0-21-generic
Found initrd image: /boot/initrd.img-6.11.0-21-generic
Found memtest86+x64 image: /boot/memtest86+x64.bin
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done

```

그림 31. 시스템 콜 테스트 전, 다음 사진처럼 부트로더를 업데이트하고 테스트를 진행해야 합니다.

#03. Taking a Glance at PCB via Syscalls

기대 결과물) syscall을 추가하는 방법과 과정이 담긴 레포트, 그리고 도중에 생성된 소스코드 파일 및 기타 추가적인 파일들 (e.g., Makefile 등)

- ~/linux-6.13.11/kernel/sys_procsched.c
- ~/linux-6.13.11/kernel/sys_procsched.o
- ~/linux-6.13.11/kernel/Makefile
- ~/linux-6.13.11/arch/x86/entry/syscalls/syscall_64.tbl
- ~/linux-6.13.11/include/linux/syscalls.h
- ~/test_sys_procsched.c

Step 1) 여러분 환경에 위에서 제시된 ‘sys_procsched’ 시스템콜을 추가하세요.



```

GNU nano 7.2 sys_procsched.c *
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/sched/signal.h>
#include <linux/types.h>
#include <linux/syscalls.h>
#include <linux/pid.h>

SYSCALL_DEFINE1(procsched, pid_t, pid) {
    struct task_struct *task;
    task = find_task_by_vpid(pid);


    if (!task)
        return -ESRCH;

#ifdef CONFIG_SCHED_INFO
    return task->sched_info.pcount;
#else
    return -ENOSYS;
#endif
}

```

그림 32. 다음과 같은 시스템 콜 구현 파일 sys_procsched.c를 생성했습니다.

→ ~/linux-6.13.11/kernel/sys_procsched.c



```

GNU nano 7.2 Makefile *
targets += config_data config_data.gz
$(obj)/config_data.gz: $(obj)/config_data FORCE
    $(call if_changed,gzip)

filechk_cat = cat $<

$(obj)/config_data: $(KCONFIG_CONFIG) FORCE
    $(call filechk,cats)

$(obj)/kheaders.o: $(obj)/kheaders_data.tar.xz

quiet_cmd_genikh = CHK      $(obj)/kheaders_data.tar.xz
      cmd_genikh = $(CONFIG_SHELL) $(srctree)/kernel/gen_kheaders.sh $@
$(obj)/kheaders_data.tar.xz: FORCE
    $(call cmd,genikh)

clean-files += kheaders_data.tar.xz kheaders.nds
obj-y += sys_hello.o
obj-y += sys_procsched.o

```

그림 33. 커널 Makefile에 sys_procsched.o를 다음과 같이 수정하였습니다.

→ ~/linux-6.13.11/kernel/Makefile

Assignment #1. Syscall

```
GNU nano 7.2                                syscall_64.tbl *
534 x32 preadv                                compat_sys_preadv64
535 x32 pwritev                               compat_sys_pwritev64
536 x32 rt_tsigqueueinfo                     compat_sys_rt_tsigqueueinfo
537 x32 recvmmsg                             compat_sys_recvmmsg_time64
538 x32 sendmmsg                             compat_sys_sendmmsg
539 x32 process_vm_readv                     sys_process_vm_readv
540 x32 process_vm_writev                   sys_process_vm_writev
541 x32 setsockopt                           sys_setsockopt
542 x32 getsockopt                           sys_getsockopt
543 x32 io_setup                             compat_sys_io_setup
544 x32 io_submit                           compat_sys_io_submit
545 x32 execveat                             compat_sys_execveat
546 x32 preadv2                             compat_sys_preadv64v2
547 x32 pwritev2                             compat_sys_pwritev64v2
548 common hello                            sys_hello
549 common procsched                        sys_procsched
# This is the end of the legacy x32 range. Numbers 548 and above are
# not special and are not to be used for x32-specific syscalls.
```

그림 34. Syscall 테이블의 549번 인덱스에 __x64_sys_procsched를 추가하였습니다.

→ ~/linux-6.13.11/arch/x86/entry/syscalls/syscall_64.tbl

```
GNU nano 7.2                                syscalls.h *
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_senedop(int semid, struct sembuf __user *tsems,
                        unsigned int nsops,
                        const struct old_timespec32 __user *timeout);
long __do_senedop(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 long __sys_hello(void);
asmlinkage long sys_procsched(pid_t pid);
#endif
```

그림 35. Syscall 헤더에 sys_procsched를 추가하였습니다.

→ ~/linux-6.13.11/include/linux/syscalls.h

```
os@os-virtual-machine:~/linux-6.13.11$ make modules -j$(nproc)
```

그림 36. 이후 커널 리빌드를 위해 모듈을 생성하였습니다.

그림과 실제 실행한 명령이 다릅니다!

- 그림: `make modules -j$(nproc)`
- 실제 코드: `make -j$(nproc)`

바로 이전 #2 과제에서는 커널 이미지를 빌드하기 위해 `make bzImage -j$(nproc)`을 사용했었습니다.

```
os@os-virtual-machine:~/linux-6.13.11$ sudo make modules_install
```

그림 37. 생성한 모듈을 설치하는 과정입니다.

```
os@os-virtual-machine:~/linux-6.13.11$ sudo make install
INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.13.11 /boot/vmlinuz-6.13.11
update-initramfs: Generating /boot/initrd.img-6.13.11
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 6.13.11 /boot/vmlinuz-6.13.11
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.13.11 /boot/vmlinuz-6.13.11
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 6.13.11 /boot/vmlinuz-6.13.11
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.13.11 /boot/vmlinuz-6.13.11
Sourcing file '/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.13.11
Found initrd image: /boot/initrd.img-6.13.11
Found linux image: /boot/vmlinuz-6.13.11.old
Found initrd image: /boot/initrd.img-6.13.11
Found linux image: /boot/vmlinuz-6.11.0-21-generic
Found initrd image: /boot/initrd.img-6.11.0-21-generic
Found memtest86+x64 image: /boot/memtest86+x64.bin
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

그림 38. 이후 커널 이미지를 설치하였습니다.

```
os@os-virtual-machine:~/linux-6.13.11$ sudo update-grub
Sourcing file '/etc/default/grub'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.13.11
Found initrd image: /boot/initrd.img-6.13.11
Found linux image: /boot/vmlinuz-6.13.11.old
Found initrd image: /boot/initrd.img-6.13.11
Found linux image: /boot/vmlinuz-6.11.0-21-generic
Found initrd image: /boot/initrd.img-6.11.0-21-generic
Found memtest86+x64 image: /boot/memtest86+x64.bin
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

그림 39. 부트로더를 업데이트하고 재부팅하였습니다.

Step 2) 잘 동작하는지 동작 확인!

```

GNU nano 7.2 test_sys_procsched.c *
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
#include <stdlib.h>

#define sys_procsched 549

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Usage: %s <pid>\n", argv[0]);
        return 1;
    }

    pid_t pid = atoi(argv[1]);
    int result = syscall(sys_procsched, pid);
    if (result < 0) {
        perror("syscall failed");
        return 1;
    }

    printf("pcount of process %d = %d\n", pid, result);
    return 0;
}

```

그림 40. 시스템 콜이 잘 만들어졌는지 테스트하기 위해 위와 같은 코드를 작성하였습니다.

→ 그림 41의 에러를 반영해서 그림은 수정하였습니다.

```

os@os-virtual-machine: $ gcc -o test_procsched user_test.c
user_test.c: In function 'main':
user_test.c:9:17: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
   9 |         printf("Usage: %s <pid> \n", argv[0]);
     |         ~~~~~
user_test.c:4:11: note: include '<stdio.h>' or provide a declaration of 'printf'
   4 |     #include <stdlib.h>
     |     ~~~~~
+++ #include <stdio.h>
   4 |
user_test.c:9:17: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
   9 |         printf("Usage: %s <pid> \n", argv[0]);
     |         ~~~~~
user_test.c:9:17: note: include '<stdio.h>' or provide a declaration of 'printf'
user_test.c:16:17: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
   16 |         printf("pcount of PID %d = %d\n", pid, ret);
     |         ~~~~~
user_test.c:16:17: note: include '<stdio.h>' or provide a declaration of 'printf'
user_test.c:18:17: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
   18 |         printf("Failed to get pcount for PID %d\n", pid);
     |         ~~~~~
user_test.c:18:17: note: include '<stdio.h>' or provide a declaration of 'printf'

```

그림 41. stdio 헤더가 없다는 에러가 발생하여 user_test.c 파일을 수정한 후 다시 컴파일을 하였습니다.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3270	os	20	0	23200	5880	3704	R	0.3	0.1	0:00.00	top
1	root	20	0	23152	13936	9328	S	0.0	0.2	0:01.25	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.02	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-kvfree_rcu_reclaim
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-sync_wq
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slub_flushwq
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns
11	root	0	-20	0	0	0	I	0.0	0.0	0:00.03	kworker/0:0H-kblockd
12	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/u512:0-ipv6_addrconf
13	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_percpu_wq
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread
16	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread
17	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
18	root	20	0	0	0	0	I	0.0	0.0	0:00.20	rcu_preempt
19	root	20	0	0	0	0	S	0.0	0.0	0:00.01	rcu_exp_par_gp_kthread_worker/1
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_exp_gp_kthread_worker
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
22	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0
23	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
24	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
25	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/1
26	root	rt	0	0	0	0	S	0.0	0.0	0:00.26	migration/1
27	root	20	0	0	0	0	S	0.0	0.0	0:00.02	ksoftirqd/1
29	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/1:0H-events_highpri
30	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/2
31	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/2
32	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	migration/2
33	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/2
34	root	20	0	0	0	0	I	0.0	0.0	0:00.37	kworker/2:0-events

그림 42. top 명령어를 통해 현재 실행 중인 프로세스와 PID를 확인할 수 있었습니다.


```

# docker exec -it ubuntu-virtual-machine: /$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         0  0.0  0.1 23152 13936 ?        Ss   00:40   0:01 /sbin/init splash
root         2  0.0  0.0      0  0 ?        S    00:40   0:00 [kthreadd]
root         3  0.0  0.0      0  0 ?        S    00:40   0:00 [pool_workqueue_release]
root         4  0.0  0.0      0  0 ?        I<   00:40   0:00 [kworker/R-kvfree_rcu_reclaim]
root         5  0.0  0.0      0  0 ?        I<   00:40   0:00 [kworker/R-rcu_gp]
root         6  0.0  0.0      0  0 ?        I<   00:40   0:00 [kworker/R-sync_wq]
root         7  0.0  0.0      0  0 ?        I<   00:40   0:00 [kworker/R-slub_flushwq]
root         8  0.0  0.0      0  0 ?        I<   00:40   0:00 [kworker/R-nets]
root        11  0.0  0.0      0  0 ?        I<   00:40   0:00 [kworker/0:0H-kblockd]
root        12  0.0  0.0      0  0 ?        I    00:40   0:00 [kworker/u512:0-tpv6_addrconf]
root        13  0.0  0.0      0  0 ?        I<   00:40   0:00 [kworker/R-mm_percpu_wq]
root        14  0.0  0.0      0  0 ?        I    00:40   0:00 [rcu_tasks_kthread]
root        15  0.0  0.0      0  0 ?        I    00:40   0:00 [rcu_tasks_rude_kthread]
root        16  0.0  0.0      0  0 ?        I    00:40   0:00 [rcu_tasks_trace_kthread]
root        17  0.0  0.0      0  0 ?        S    00:40   0:00 [ksoftirqd/0]
root        18  0.0  0.0      0  0 ?        I    00:40   0:00 [rcu_preempt]
root        19  0.0  0.0      0  0 ?        S    00:40   0:00 [rcu_exp_par_gp_kthread_worker/1]
root        20  0.0  0.0      0  0 ?        S    00:40   0:00 [rcu_exp_gp_kthread_worker]
root        21  0.0  0.0      0  0 ?        S    00:40   0:00 [migration/0]
root        22  0.0  0.0      0  0 ?        S    00:40   0:00 [idle_inject/0]
root        23  0.0  0.0      0  0 ?        S    00:40   0:00 [cpuhp/0]
root        24  0.0  0.0      0  0 ?        S    00:40   0:00 [cpuhp/1]
root        25  0.0  0.0      0  0 ?        S    00:40   0:00 [idle_inject/1]
root        26  0.0  0.0      0  0 ?        S    00:40   0:00 [migration/1]
root        27  0.0  0.0      0  0 ?        S    00:40   0:00 [ksoftirqd/1]

```

그림 43. ps aux 명령어를 통해 현재 실행 중인 모든 프로세스들의 정보를 확인할 수 있었습니다.

```
os@os-virtual-machine:~$ ./test_sys_procsched 1
syscall failed: Function not implemented
```

그림 44. gcc -o test_sys_procsched test_sys_procsched.c를 통해 컴파일한 후 시스템 콜 함수가 정상적으로 호출되도록 연결됐는지 확인하기 위해 실행한 결과 **syscall 호출에 실패했다는 에러**를 마주했습니다.

```
os@os-virtual-machine:~$ ./test_sys_procsched 1
syscall failed: Function not implemented
os@os-virtual-machine:~$ strace ./test_sys_procsched 1
execve("./test_sys_procsched", ["/test_sys_procsched", "1"], 0x7ffe7d4cb8a8 /* 48 vars */) = 0
brk(NULL)                               = 0x64e282d51000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x734335d4f000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=57123, ...}) = 0
mmap(NULL, 57123, PROT_READ, MAP_PRIVATE, 3, 0) = 0x734335d41000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0-\0\1\0\0\0\220\243\2\0\0\0\0....", 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0... ", 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0...", 784, 64) = 784
mmap(NULL, 2178256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x734335a00000
mmap(0x734335a28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x734335a28000
mmap(0x734335b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x734335b00000
mmap(0x734335bff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x734335bff000
mmap(0x734335c50000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x734335c50000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x734335d3e000
arch_prctl(ARCH_SET_FS, 0x734335d3e740) = 0
set_tid_address(0x734335d3ea10)         = 3008
set_robust_list(0x734335d3ea20, 24)     = 0
rseq(0x734335d3f060, 0x20, 0, 0x53053053) = 0
mprotect(0x734335bff000, 16384, PROT_READ) = 0
mprotect(0x64e282d207000, 4096, PROT_READ) = 0
mprotect(0x734335d8d000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x734335d41000, 57123)           = 0
syscall_0x225(0x1, 0, 0x7ffe6493d258, 0x1999999999999999, 0, 0x7ffe6493c118) = -1 ENOSYS (Function not implemented)
dup(2)                                   = 3
fcntl(3, F_GETFL)                       = 0x80002 (flags O_RDWR|O_CLOEXEC)
getrandom("\xf2\x9b\xa0\xee\x42\x2c\xad\x66", 8, GRND_NONBLOCK) = 0
brk(NULL)                               = 0x64e282d51000
brk(0x64e282d72000)                    = 0x64e282d72000
fstat(3, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(3, "syscall failed: Function not imp..., 41syscall failed: Function not implemented\n") = 41
close(3)                                = 0
exit_group(1)                          = ?
+++ exited with 1 +++
```

그림 45. strace로 syscalls의 실제 호출이 어디로 가는지 확인해 보았습니다.

syscall_0x225(0x1, 0x7ffe6493c118) = -1 ENOSYS (Function not implemented)

- syscall 호출 자체가 시스템 콜 테이블에 등록되지 않은 상태를 의미합니다.
- 0x225: 10진수로 549번, 즉 등록된 시스템 콜 번호를 의미합니다.
- ENOSYS: 시스템 콜 테이블에 등록되지 않았거나, 빌드에 포함되지 않았다는 의미입니다.

위 에러 발생 시 확인이 필요한 부분은 다음과 같습니다.

- ~/linux-6.13.11/arch/x86/syscalls/syscall_64.tbl - 확인 결과 오류 없음
- ~/linux-6.13.11/kernel/sys_procsched.c - 확인 결과 오류 없음
- ~/linux-6.13.11/kernel/Makefile - 확인 결과 오류 없음

이후 시스템 콜 확인을 위해 grep procsched /proc/kallsyms를 통해 시스템 콜 함수가 커널에 제대로 포함되어 있는지 확인하는 절차를 거쳤습니다.

```
os@os-virtual-machine:~$ grep procsched /proc/kallsyms
os@os-virtual-machine:~$
```

그림 46. 아무것도 나오지 않았다? → 정의된 함수가 커널 이미지에 포함되지 않은 상태이다.

```
os@os-virtual-machine:~/linux-6.13.11$ cp /boot/config-$(uname -r) .config
os@os-virtual-machine:~/linux-6.13.11$ make olddefconfig
HOSTCC scripts/kconfig/conf.o
HOSTLD scripts/kconfig/conf
#
# No change to .config
#
os@os-virtual-machine:~/linux-6.13.11$ make clean
CLEAN arch/x86/boot/compressed
CLEAN arch/x86/boot
CLEAN arch/x86/crypto
CLEAN arch/x86/entry/vdso
CLEAN arch/x86/kernel/cpu
CLEAN arch/x86/kernel
CLEAN arch/x86/kvm
CLEAN arch/x86/purgatory
CLEAN arch/x86/realmode/rm
CLEAN arch/x86/tools
CLEAN arch/x86/lib
CLEAN certs
CLEAN crypto/asymmetric_keys
CLEAN drivers/accessibility/speakup
CLEAN drivers/firmware/efi/libstub
CLEAN drivers/gpu/drm/radeon
CLEAN drivers/gpu/drm/xe
CLEAN drivers/net/wan
CLEAN drivers/scsi/aic7xxx
CLEAN drivers/scsi
CLEAN drivers/tty/vt
CLEAN fs/unicode
CLEAN init
CLEAN kernel/debug/kdb
CLEAN kernel
CLEAN lib/raid6
CLEAN lib/test_fortify
CLEAN lib
CLEAN net/wireless
CLEAN security/apparmor
CLEAN security/selinux
CLEAN security/tomoyo
CLEAN usr
CLEAN
CLEAN vmlinux.symvers modules.builtin modules.builtin.modinfo .vmlinux.objs .vmlinux.export.c
```

그림 47. make install 후 /boot 에 생긴 .config 파일의 문제가 있을 수도 있다고 생각이 들어 다른 작업도 진행해 보았으나, 여기 또한 다른 문제는 없었으며 커널을 완전히 초기화시키고 다시 빌드를 해보았습니다.

```
os@os-virtual-machine:~$ grep procsched /proc/kallsyms
0000000000000000 T __pfx__x64_sys_procsched
0000000000000000 T __x64_sys_procsched
0000000000000000 T __pfx__ia32_sys_procsched
0000000000000000 T __ia32_sys_procsched
0000000000000000 d event_exit__procsched
0000000000000000 d event_enter__procsched
0000000000000000 d __syscall_meta__procsched
0000000000000000 d args__procsched
0000000000000000 d types__procsched
0000000000000000 d __event_exit__procsched
0000000000000000 d __event_enter__procsched
0000000000000000 d __p_syscall_meta__procsched
0000000000000000 d __eil_addr__ia32_sys_procsched
0000000000000000 d __eil_addr__x64_sys_procsched
```

그림 48. grep procsched /proc/kallsyms를 통해 커널에 sys_procsched가 포함되어있는지 확인한 결과, 포함되어 있었습니다!

```
os@os-virtual-machine:~$ gcc -o test_sys_procsched test_sys_procsched.c
os@os-virtual-machine:~$ ./test_sys_procsched 1234
pcount of process 1234 = 1

os      2443  0.0  0.0  19692  4968 pts/0    Ss   21:44  0:00  bash
root    2451  0.0  0.0      0   0 ?        I    21:44  0:00  [kworker/u516:0-kvfree_rcu_reclaim]
root    2452  0.0  0.0      0   0 ?        I    21:44  0:00  [kworker/u515:1-events_unbound]
root    2453  0.0  0.0      0   0 ?        I    21:44  0:00  [kworker/2:1-cgroup_destroy]
root    2459  0.0  0.0      0   0 ?        I<   21:44  0:00  [kworker/u518:4-ttn]
root    2460  0.0  0.0      0   0 ?        I<   21:44  0:00  [kworker/u519:4]
root    2461  0.0  0.0      0   0 ?        I<   21:44  0:00  [kworker/u519:5]
root    2462  0.0  0.0      0   0 ?        I<   21:44  0:00  [kworker/u521:6]
root    2463  0.0  0.0      0   0 ?        I<   21:44  0:00  [kworker/u521:7]
root    2473  0.0  0.0      0   0 ?        I    21:46  0:00  [kworker/u513:2-flush-8:0]
root    2478  0.0  0.0      0   0 ?        I    21:48  0:00  [kworker/u514:1-flush-8:0]
root    2500  0.0  0.0      0   0 ?        I    22:02  0:00  [kworker/u513:0-events_unbound]
root    2509  0.0  0.0      0   0 ?        I    22:10  0:00  [kworker/1:0-events_freezable]
root    2510  0.0  0.0      0   0 ?        I    22:10  0:00  [kworker/u513:1-events_unbound]
root    2516  0.0  0.0      0   0 ?        I    22:12  0:00  [kworker/u514:0-events_unbound]
os      2636 100  0.0  22204  4600 pts/0    R+   22:17  0:00  ps aux
os@os-virtual-machine:~$ ./test_sys_procsched 2480
syscall failed: No such process
os@os-virtual-machine:~$ ./test_sys_procsched 2443
pcount of process 2443 = 310
os@os-virtual-machine:~$ ./test_sys_procsched 2451
pcount of process 2451 = 380
```

그림 49-50. 이후 테스트 코드를 컴파일하여 정상적으로 pcount 값이 출력되는 것을 확인하였습니다.