

# assignment 01

인공지능학부

211852

조나현

# 01. compiling linux kernel

# 커널 소스 다운

mkdir exercise

cd exercise

wget <https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.19.12.tar.xz>

tar xvf linux-5.19.12.tar.xz

==> exercise 폴더를 만들고 그 폴더 안에 커널 소스를 다운받아서 압축을 해제한다.

# 01. compiling linux kernel

# 필요한 프로그램 (tool) 설치

```
sudo apt-get update && sudo apt-get upgrade -y
```

```
sudo apt-get install build-essential libncurses-dev libssl-dev libelf-dev bison flex -y
```

#커널 설정

```
cd linux-5.19.12
```

```
cp /boot/config-5.19.0-41-generic .config
```

```
make menuconfig
```

```
scripts/config --disable SYSTEM_TRUSTED_KEYS #to avoid the certification check error
```

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

## 01. compiling linux kernel

# 컴파일

nproc ==> 4가 나와서 코어의 개수가 4임을 확인

make -j4

#커널 설치

sudo make modules\_install -j4

sudo make install -j4

## 01. compiling linux kernel

#부트로더 업데이트

```
sudo update-grub
```

#커널 버전 확인

```
sudo reboot
```

```
uname -a
```

```
cat /etc/os-release
```

# 01. compiling linux kernel

## #결과물

```
jonahyun@jonahyun-virtual-machine: ~  
jonahyun@jonahyun-virtual-machine:~$ uname -a  
Linux jonahyun-virtual-machine 5.19.12 #1 SMP PREEMPT_DYNAMIC Thu May 4 18:59:19  
KST 2023 x86_64 x86_64 x86_64 GNU/Linux  
jonahyun@jonahyun-virtual-machine:~$ cat /etc/os-release  
PRETTY_NAME="Ubuntu 22.04.2 LTS"  
NAME="Ubuntu"  
VERSION_ID="22.04"  
VERSION="22.04.2 LTS (Jammy Jellyfish)"  
VERSION_CODENAME=jammy  
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=jammy  
jonahyun@jonahyun-virtual-machine:~$
```

## 02. Adding my first system calls

#시스템 콜 함수 구현

cd linux-5.19.12/kernel

vi hello.c

```
jonahyun@jonahyun-virtual-machine: ~/linux-5.19.12/kernel
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE0(hello)
{
    printk("hello, nahyun\n");
    printk("211852\n");
    return 0;
}
```

## 02. Adding my first system calls

#시스템 콜의 makefile에 등록

cd linux-5.19.12/kernel/Makefile

```
Jonahyun@Jonahyun-virtual-machine: ~/linux-5.19.12/kernel
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 platform-feature.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 hello.o
```



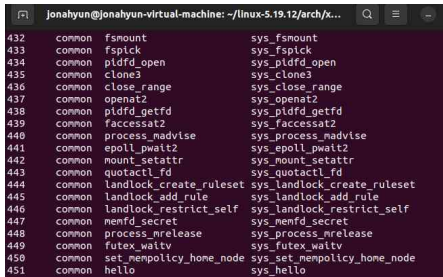
## 02. Adding my first system calls

# 시스템 콜 테이블에 등록

cd linux-5.19.12/arch/x86/entry/syscalls

vi syscall\_64.tbl

==>451번에 등록함



```
Jonahyun@Jonahyun-virtual-machine: ~/linux-5.19.12/arch/x...
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_huge_node sys_set_mempolicy_huge_node
451 common hello sys_hello
```

## 02. Adding my first system calls

# 시스템 콜 헤더 파일에 등록

cd linux-5.4.59/include/linux

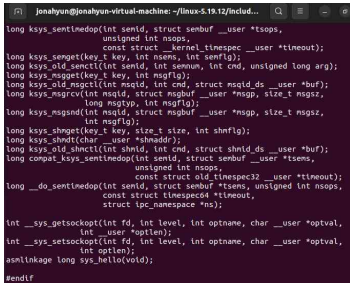
vi syscalls.h

해당 파일의 마지막에 있는 #endif 앞에

다음과 같이

자신의 함수 정의에 맞게 작성

==> `asm linkage long sys_hello(void);`



```
Jonahyun@Jonahyun-virtual-machine: ~/linux-5.19.12/includ...
long ksys_sengetimedop(int senid, struct senbuf __user *tsops,
    unsigned int nsops,
    const struct __kernel_timespec __user *timeout);
long ksys_senget(key_t key, int nsens, int senflg);
long ksys_old_senctl(int senid, 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_shmctl(char __user *shmaddr);
long ksys_old_shmctl(int shmid, int cmd, struct shmid_ds __user *buf);
long ksys_compat_ksys_sengetimedop(int senid, struct senbuf __user *tsens,
    unsigned int nsops,
    const struct old_timespec32 __user *timeout);
long __do_sengetimedop(int senid, struct senbuf *tsens, 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);
asm linkage long sys_hello(void);

#endif
```

## 02. Adding my first system calls

# 커널 컴파일 및 재부팅

sudo make -j4

sudo make install

sudo update-grub

sudo reboot

## 02. Adding my first system calls

```
#test
```

[테스트 코드] -> vi test2.c로 저장

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

```
#include <sys/syscall.h>
```

```
#define MY_SYS_HELLO 451
```

```
int main(int argc, char *argv[]) {  
    int ret = syscall(MY_SYS_HELLO);  
    return 0;  
}
```

## 02. Adding my first system calls

### #결과물

=>gcc test.c 와 ./a.out 라는 명령어를 실행 한후 확인 가능

<dmesg 명령어>

```
[ 34.201855] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[ 35.524988] loop11: detected capacity change from 0 to 8
[ 51.186461] audit: type=1326 audit(1683444294.768:47): auid=1000 uid=1000 gid=1000 ses=2 subj=snap.snapd-desktop-integration.snapd-desktop-integration(enforce) pid=1354 comm="snapd-desktop-i" exe="/snap/snapd-desktop-integration/49/usr/bin/snapd-desktop-integration" sig=0 arch=c000003e syscall=314 compat=0 ip=0x7f7151f3b73d code=0x50000
[ 52.251917] rfkill: input handler disabled
[ 329.034849] hello, nahyun
[ 329.034858] 211852
```

<cat /dev/kmsg 명령어>

```
6,1941,34201855,-;IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
6,1942,35524988,-;loop11: detected capacity change from 0 to 8
5,1943,51186461,-;audit: type=1326 audit(1683444294.768:47): auid=1000 uid=1000 gid=1000 ses=2 subj=snap.snapd-desktop-integration.snapd-desktop-integration(enforce) pid=1354 comm="snapd-desktop-i" exe="/snap/snapd-desktop-integration/49/usr/bin/snapd-desktop-integration" sig=0 arch=c000003e syscall=314 compat=0 ip=0x7f7151f3b73d code=0x50000
7,1944,52251917,-;rfkill: input handler disabled
4,1945,329034849,-;hello, nahyun
4,1946,329034858,-;211852
```

## 03 . Taking a glance at PCB via Syscalls

#시스템 콜 함수 구현

cd linux-5.19.12/kernel

vi procsched.c

```
#include <linux/kernel.h>
#include <linux/syscalls.h>
#include <linux/linkage.h>
#include <unistd.h>
#include <stdio.h>
#include <linux/sched.h>
#include <linux/types.h>

SYSCALL_DEFINE1 (procsched, pid)
{
    struct task_struct *task;
    struct sched_info *sched;


    task = find_task_by_vpid(pid);
    if (task == NULL){
        return -ESRCH;
    }

    sched = &task->sched_info;
    return sched->pcount;
}
```

## 03 . Taking a glance at PCB via Syscalls

#시스템 콜의 makefile에 등록

cd linux-5.19.12/kernel/Makefile

A terminal window with a dark purple background. The title bar shows the user 'jonahyun' on a 'jonahyun-virtual-machine' at the directory '~/linux-5.19.12/kernel'. The terminal content shows the beginning of the 'Makefile' for the Linux kernel, including the SPDX license identifier and a list of object files for the 'obj-y' target.

```
jonahyun@jonahyun-virtual-machine: ~/linux-5.19.12/kernel
## 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 platform-feature.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 hello.o procsched.o
```

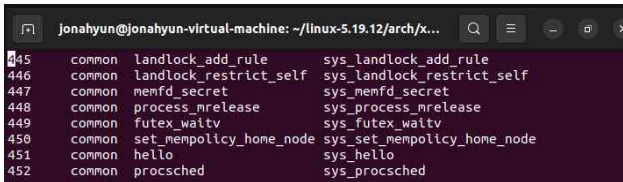
## 03 . Taking a glance at PCB via Syscalls

# 시스템 콜 테이블에 등록

cd linux-5.19.12/arch/x86/entry/syscalls

vi syscall\_64.tbl

==>452번에 등록함



```
jonahyun@jonahyun-virtual-machine: ~/linux-5.19.12/arch/x...  
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_release sys_process_release  
449 common futex_waitv sys_futex_waitv  
450 common set_mempolicy_home_node sys_set_mempolicy_home_node  
451 common hello sys_hello  
452 common procsched sys_procsched
```



## 03 . Taking a glance at PCB via Syscalls

# 시스템 콜 헤더 파일에 등록

linux-5.4.59/include/linux

vi syscalls.h

해당 파일의 마지막에 있는 #endif 앞에

다음과 같이

자신의 함수 정의에 맞게 작성

=>asmlinkage long sys\_procsched(procsched, pid);

```
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(procsched, pid);
```

## 03 . Taking a glance at PCB via Syscalls

# 커널 컴파일 및 재부팅

```
sudo make -j4
```

```
sudo make install
```

```
sudo update-grub
```

```
sudo reboot
```

=> 2번을 해결하는데 3일 이상이 소요되어 마지막 문제 컴파일을 과제제출 4시간전에 수행했으나 컴파일이 완료되지 않았습니다.. (평균 컴파일 시간이 5시간,,)

## 03 . Taking a glance at PCB via Syscalls

#test

<테스트 코드>

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

```
#include <stdio.h>
```

```
#include <sys/syscall.h>
```

```
#define MY_SYS_PROCSCHED 548 // 추가한 syscall의 인덱스가 548이라면...
```

```
int main(int argc, char *argv[]) {
```

```
    int ret = syscall(MY_SYS_PROCSCHED, 1234); //PID 1234
```

```
    printf("pcount of 1234 = %d \n", ret);
```

```
    return 0;
```

```
}
```

## 03 . Taking a glance at PCB via Syscalls

#결과물

결과물 도출을 실패하였다.