01. Compiling Linux kernel

1. 커널 코드를 다운로드하고 압축을 푼다.

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

2. 커널 컴파일에 필요한 것들을 설치한다.

```
sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev
bc flex libelf-dev bison
```

2. 커널 파일을 이동한 후 현재 위치도 그 곳으로 이동한다.

```
mv linux-5.19.12 /usr/src/
cd /usr/src/linux-5.19.12
```

3. .config을 파일을 기본 커널 소스에서 복사해 가져온다.

```
cp /boot/config-$(uname -r) .config
```

4. .config 파일의 두 키를 다음과 같이 수정해 준다.

```
CONFIG_SYSTEM_TRUSTED_KEYS=“”
CONFIG_SYSTEM_REVOCATION_KEYS=“”
```

5. 사용할 모듈을 선택한다.

```
make menuconfig
```

6. 커널을 컴파일 하고 설치한다.

```
make -j4
sudo make modules_install
sudo make install
```

7. 설치한 커널로 변경해본다.

```
reboot
```

8. 변경된 커널 정보를 확인한다.

```
uname -a
cat /etc/os-release
```

```
mtak@mtak-virtual-machine:~$ uname -a
Linux mtak-virtual-machine 5.19.12 #1 SMP PREEMPT_DYNAMIC Sat Apr 29 21:25:30 KST 2023 x86_64 x86_64 x86_64 GNU/Linux
mtak@mtak-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
mtak@mtak-virtual-machine:~$ s
```

02. Adding my first system calls

1. 직접 만들 system call을 저장할 파일을 만든다.

```
mkdir /usr/src/linux-5.19.12/identity
```

2. 시스템 콜 함수를 구현한다.

```
// path: /usr/src/linux-5.9.12/identity/hello.c
#include <linux/kernel.h>
#include <linux/syscalls.h>
SYSCALL_DEFINE0(hello) {
        printk("Hello, MinKyeong.Tak\n");
        printk("192293\n");
        return 0;
}
```

3. 구현한 시스템 콜을 위한 Makefile을 만든다.

```
# path: /usr/src/linux-5.9.12/identity/Makefile
obj-y := hello.o
```

4. 구현 시스템 함수들의 홈 디렉토리(identity) 를 메인 Makefile에 등록한다.

```
# path: /usr/src/linux-5.9.12/Makefile
kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ identity/
```

5. 시스템 콜의 헤더에 만든 함수의 원형을 등록한다

```
// path: /usr/src/linux-5.9.12/include/linux/syscalls.h
asmlinkage long sys_hello(void);
#endif
```

6. 시스템콜 테이블에 등록한다

```
#  path: /usr/src/linux-5.19.12/arch/x86/entry/syscalls/syscall_64.tbl
451 common  hello           sys_hello
```

7. 커널을 컴파일하고 설치한다.

```
make
sudo make install
```

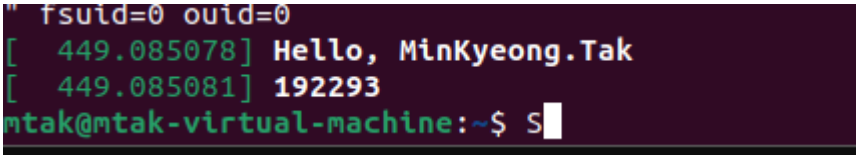
8. os의 부트 로더를 새로운 커널로 업데이트 한다.

```
sudo update-grub
```

9. 재부팅 한다

```
reboot
```

10. 예제 파일을 실행해본다.



03. Taking a glance at PCB via Syscalls

1. /identity 에 새로운 시스템 콜을 작성한다.

```
#include <linux/kernel.h>
#include <linux/sched.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE1(get_pcount, int pid)
{
    struct task_struct *task;
    struct sched_info si;

    task = pid_task(find_vpid(pid), PIDTYPE_PID); // 해당 PID에 대한
task_struct 구조체 가져오기
    if (!task)
        return -ESRCH;

    rcu_read_lock(); // read-side lock을 획득
    si = task->sched_info; // task_struct 내부의 sched_info 구조체 가져오기
    rcu_read_unlock(); // read-side lock 해제

    return si.pcount; // pcount 값 반환
}
```

2. 구현한 시스템콜을 추가한다.

```
# path: /usr/src/linux-5.9.12/identity/Makefile
obj-y := hello.o procsched.o
```

3. 시스템 콜의 헤더에 만든 함수의 원형을 등록한다

```
// path: /usr/src/linux-5.9.12/include/linux/syscalls.h
asmlinkage long sys_hello(void);
asmlinkage long sys_procsched(int);
#endif
```

4. 시스템콜 테이블에 등록한다

```
#  path: /usr/src/linux-5.19.12/arch/x86/entry/syscalls/syscall_64.tbl
451 common  hello           sys_hello
452 common  procsched       sys_procsched
```

5. 커널을 컴파일하고 설치한다.

```
make
sudo make install
```

6. os의 부트 로더를 새로운 커널로 업데이트 한다.

```
sudo update-grub
```

7. 재부팅 한다

```
reboot
```

8. 예제 파일을 실행해본다.