

# 운영체제 Assignment 1

이름 : 이준휘

학번 : 2018202046

교수 : 최상호 교수님

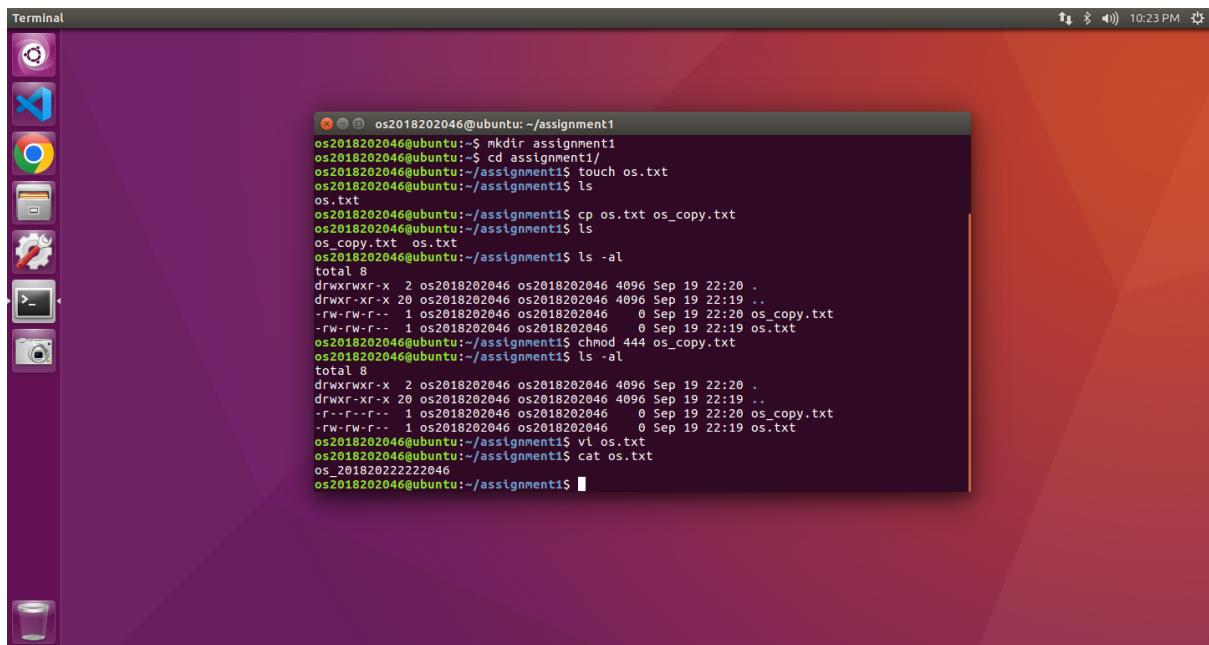
강의 시간 : 금 1, 2

## 1. Introduction

해당 과제는 다음과 같은 3개의 소과제로 나눈다. 첫 번째 과제에서는 Linux의 Basic Command에 대하여 확인하는 과정을 거친다. 그 후 두 번째 과제에서는 Kernel 4.19.67을 Compile하는 과정을 통해 kernel을 어떻게 Compile하는지를 익힌다. 마지막으로 세 번째 과제에서는 Linux app가 실행되는 지점 탐색을 진행하면서 탐색 방법과 커널 메시지 출력 방법에 대해 실습한다.

## 2. Conclusion & Analysis

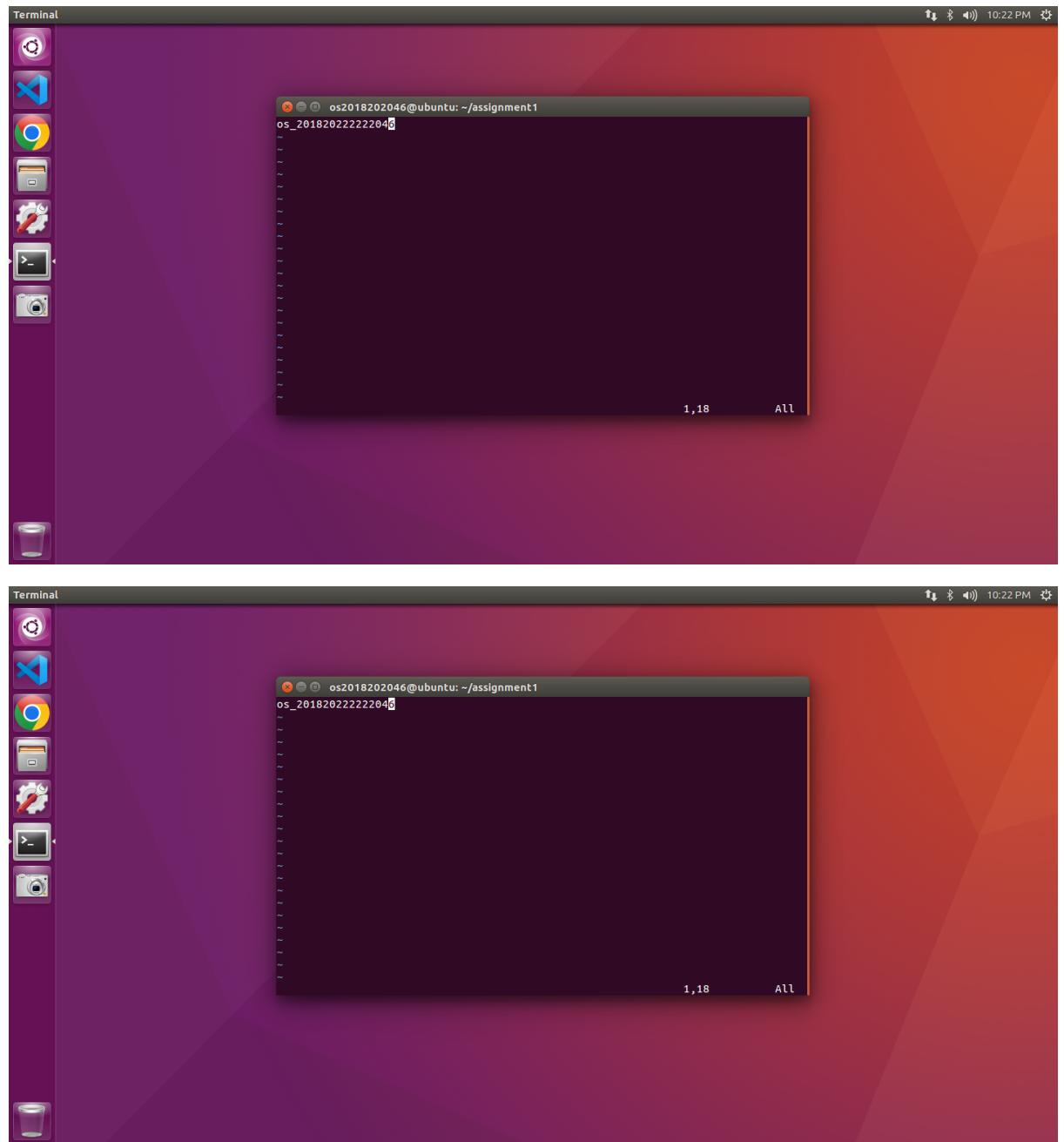
### A. Assignment 1-1



```
Terminal
os2018202046@ubuntu:~/assignment1
os2018202046@ubuntu:~$ mkdir assignment1
os2018202046@ubuntu:~$ cd assignment1/
os2018202046@ubuntu:~/assignment1$ touch os.txt
os2018202046@ubuntu:~/assignment1$ ls
os.txt
os2018202046@ubuntu:~/assignment1$ cp os.txt os_copy.txt
os2018202046@ubuntu:~/assignment1$ ls
os_copy.txt os.txt
os2018202046@ubuntu:~/assignment1$ chmod 444 os_copy.txt
os2018202046@ubuntu:~/assignment1$ ls -al
total 8
drwxrwxr-x 2 os2018202046 os2018202046 4096 Sep 19 22:20 .
drwxr-xr-x 20 os2018202046 os2018202046 4096 Sep 19 22:19 ..
-rw-rw-r-- 1 os2018202046 os2018202046 0 Sep 19 22:20 os_copy.txt
-rw-rw-r-- 1 os2018202046 os2018202046 0 Sep 19 22:19 os.txt
os2018202046@ubuntu:~/assignment1$ vi os.txt
os2018202046@ubuntu:~/assignment1$ cat os.txt
os_201820222222046
os2018202046@ubuntu:~/assignment1$
```

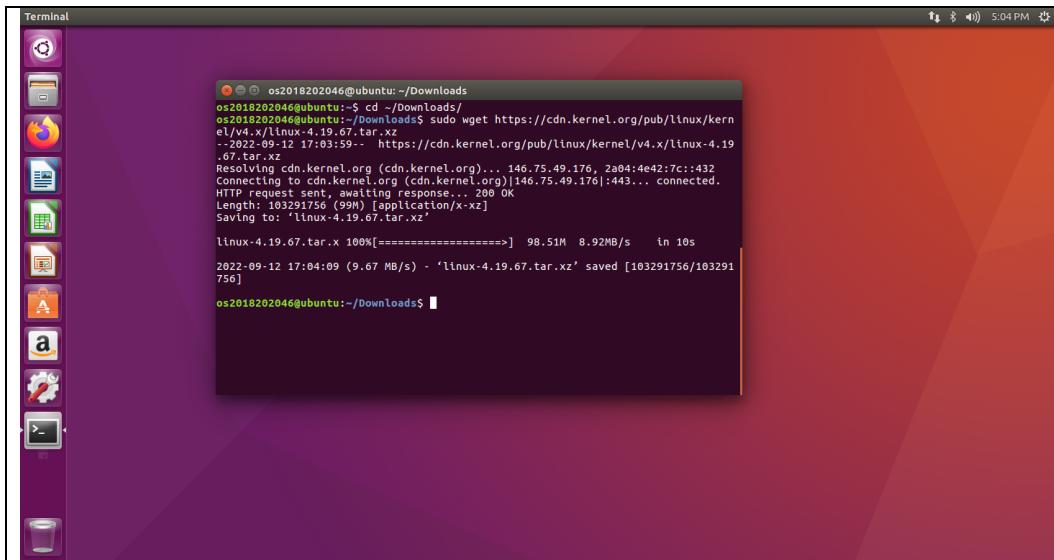
해당 사진은 1-1 과제를 수행한 모습이다. 우선 username이 os2018202046으로 필자의 학번에 맞게 생성되었다. 그리고 다음은 과제에서 빈 칸에 맞는 명령을 수행한 결과다. 우선 1번으로 assignment1명의 디렉터리를 생성하기 위해 mkdir명령을 사용하였다. 그 후 2번으로 assignment1 디렉터리로 이동하기 위해 cd assignment1을 통해 이동 후 "os.txt"파일을 만들기 위해 touch os.txt를 입력하였다. ls를 입력하였을 때 해당 파일은 정상적으로 생성되었다. 3번에서는 os.txt를 os\_copy.txt명으로 복사하기 위해 cp os.txt os\_copy.txt 명령을 사용하였다. 그 결과 해당 파일명으로 파일이 복사되었다. 4번에서는 os\_copy.txt의 권한을 모든 대상에게 읽기만 부여하는 것으로 이를 위해 chmod 444 os\_copy.txt를 사용하여 4(읽기 권한)만 부여하였다. 그 결과 다음 줄에서 ls -al로 권한을 확인하였을 때 모두 읽기 권한만 존재하는 것을 알 수 있었다. 마지막으로 5번에서는 os.txt에 os\_학번을 작성 후 터미널에 출력하는 것이다. 우선

os.txt에 학번을 작성하기 위해 vi os.txt를 사용한다. 그 후 i를 입력하여 insert 모드로 진입 후 os\_2018202046을 입력한다. 입력을 마친 후에는 insert모드에서 나오기 위해 esc를 입력하고 :wq를 입력하여 저장 및 vi를 종료한다. 그 후 cat os.txt를 입력하여 os.txt를 출력하였을 때 다음과 같이 정상적으로 출력되는 것을 확인할 수 있다.



## B. Assignment 1-2

해당 과제는 우분투 내에서 kernel을 다운로드하고, 컴파일의 모든 과정을 terminal과 vi를 통해 진행한다.

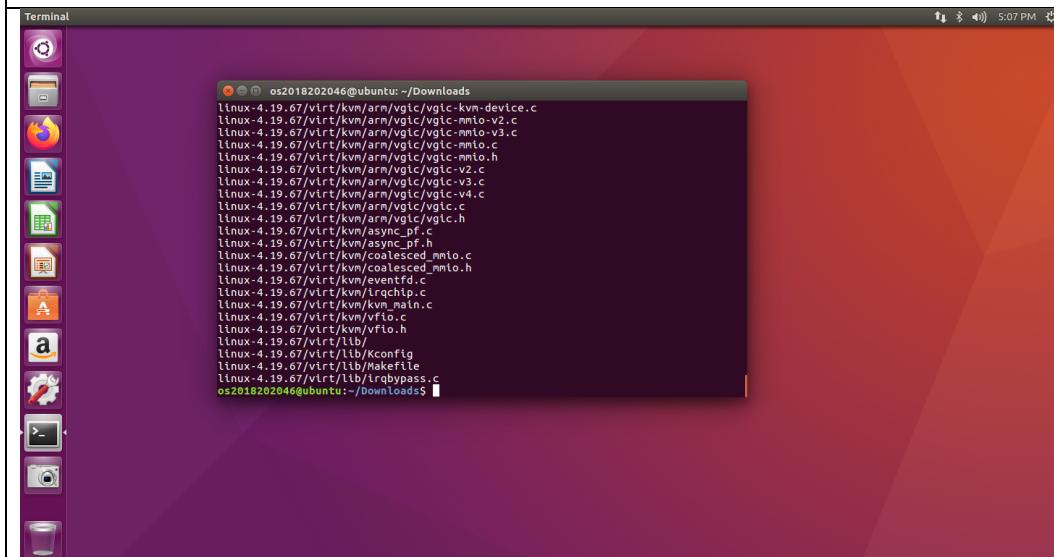


```
os@os2018202046:~/Downloads$ cd ~/Downloads/
os@os2018202046:~/Downloads$ sudo wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.19.67.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 146.75.49.176, 2a04:4e42:7c::432
Connecting to cdn.kernel.org (cdn.kernel.org)|146.75.49.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 103291756 (99M) [application/x-xz]
Saving to: 'linux-4.19.67.tar.xz'

linux-4.19.67.tar.x 100%[=====] 98.51M 8.92MB/s   in 10s
2022-09-12 17:04:09 (9.67 MB/s) - 'linux-4.19.67.tar.xz' saved [103291756/103291756]

os@os2018202046:~/Downloads$
```

우선 Downloads 디렉토리로 이동하여 sudo wget 명령을 통해 linux-4.19.67 커널을 다운받는다.



```
os@os2018202046:~/Downloads$ tar -Jxvf linux-4.19.67.tar.xz
linux-4.19.67/virt/kvm/arm/vgic/vgtc-kvm-device.c
linux-4.19.67/virt/kvm/arm/vgic/vgtc-mmio-v2.c
linux-4.19.67/virt/kvm/arm/vgic/vgtc-mmio-v3.c
linux-4.19.67/virt/kvm/arm/vgic/vgtc-mmio.h
linux-4.19.67/virt/kvm/arm/vgic/vgtc-mmio_mmio.c
linux-4.19.67/virt/kvm/arm/vgic/vgtc-mmio_mmio.h
linux-4.19.67/virt/kvm/arm/vgic/vgtc-v2.c
linux-4.19.67/virt/kvm/arm/vgic/vgtc-v3.c
linux-4.19.67/virt/kvm/arm/vgic/vgtc-v4.c
linux-4.19.67/virt/kvm/arm/vgic/vgtc.c
linux-4.19.67/virt/kvm/arm/vgic/vgtc.h
linux-4.19.67/virt/kvm/arm/vsync_pt.c
linux-4.19.67/virt/kvm/arm/vsync_pt.h
linux-4.19.67/virt/kvm/coalesced_mmio.c
linux-4.19.67/virt/kvm/coalesced_mmio.h
linux-4.19.67/virt/kvm/eventfd.c
linux-4.19.67/virt/kvm/irqchip.c
linux-4.19.67/virt/kvm/kvm_main.c
linux-4.19.67/virt/kvm/vfio.c
linux-4.19.67/virt/kvm/vfio.h
linux-4.19.67/virt/lib/Kconfig
linux-4.19.67/virt/lib/Makefile
linux-4.19.67/virt/lib/lrqbypass.c
os@os2018202046:~/Downloads$
```

다운받은 커널은 tar -Jxvf 명령을 통해 압축을 풀어준다

The image shows a dual-terminal setup on an Ubuntu desktop. The top terminal window displays the contents of the `Makefile` for the Linux kernel version 4.19.67. It includes configuration details like `SPDX-License-Identifier: GPL-2.0`, `PATCHLEVEL = 19`, and the `EXTRAVERSION = -2018202046`. The bottom terminal window shows the user navigating to the kernel source directory and listing its contents, which include the `Linux-4.19.67.tar.xz` archive and the `Makefile`.

```
# SPDX-License-Identifier: GPL-2.0
VERSION = 4.19
PATCHLEVEL = 19
SUBLEVEL = 67
EXTRAVERSION = -2018202046
NAME = "People's Front"

# *DOCUMENTATION*
# To see a list of typical targets execute "make help"
# More info can be located in /README
# Comments in this file are targeted only to the developer, do not
# expect to learn how to build the kernel reading this file.

# That's our default target when none is given on the command line
PHONY := _all
_all:
    # o Do not use make's built-in rules and variables
    #   (this increases performance and avoids hard-to-debug behaviour);
    # o Look for make include files relative to root of kernel src
    MAKEFLAGS += -rR -include-dir=$(CURDIR)

# Avoid funny character set dependencies
5,26          Top
```

```
os2018202046@ubuntu: ~/Downloads/linux-4.19.67$ ls
Linux-4.19.67.tar.xz
os2018202046@ubuntu:~/Downloads$ cd Linux-4.19.67/
os2018202046@ubuntu:~/Downloads/Linux-4.19.67$ vi Makefile
os2018202046@ubuntu:~/Downloads/Linux-4.19.67$
```

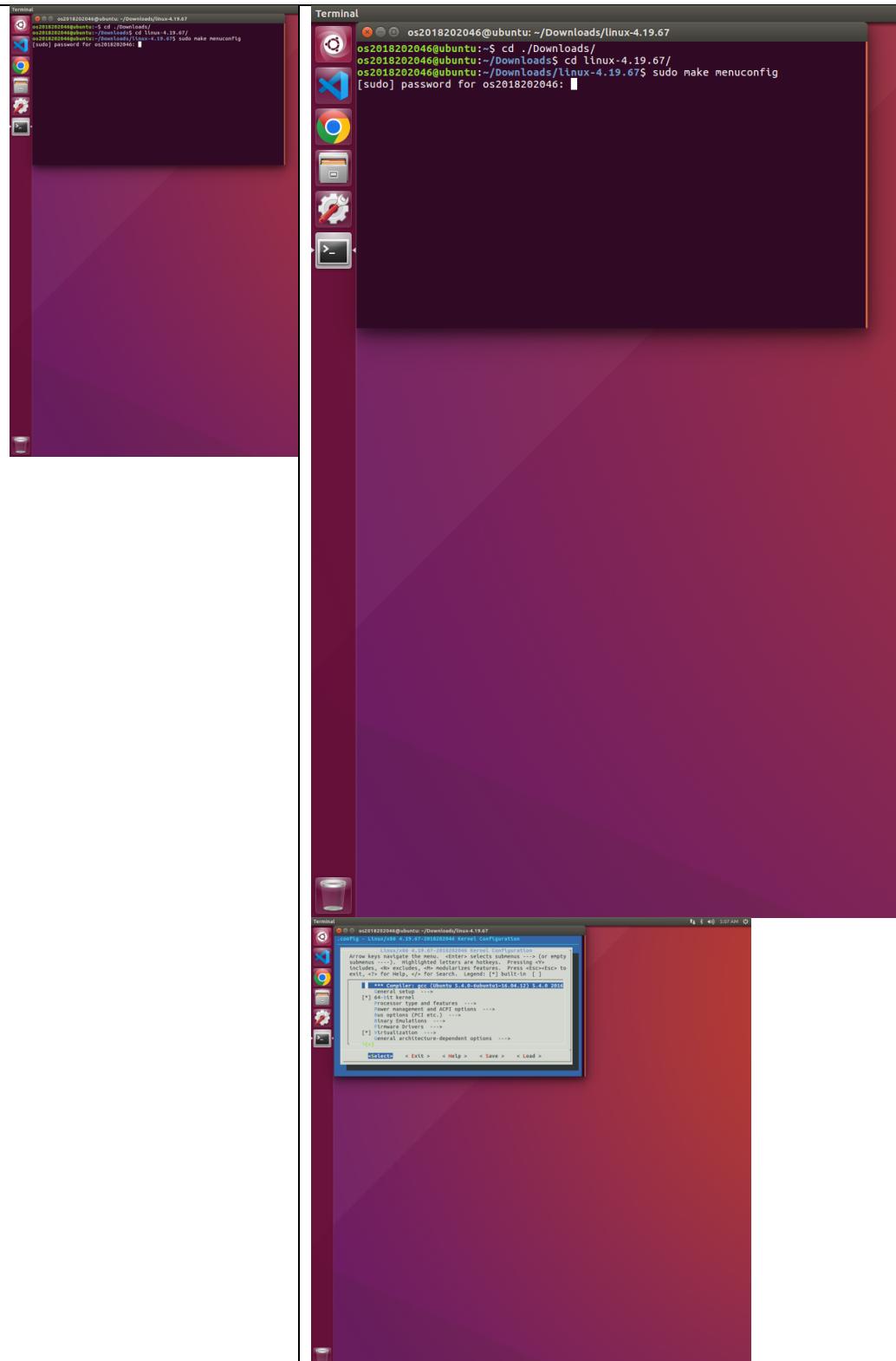
Kernel의 Extra Version을 수정하기 위해 Makefile을 vim editor로 열어 EXTRAVERSION에 -2018202046(학번)을 입력한다.

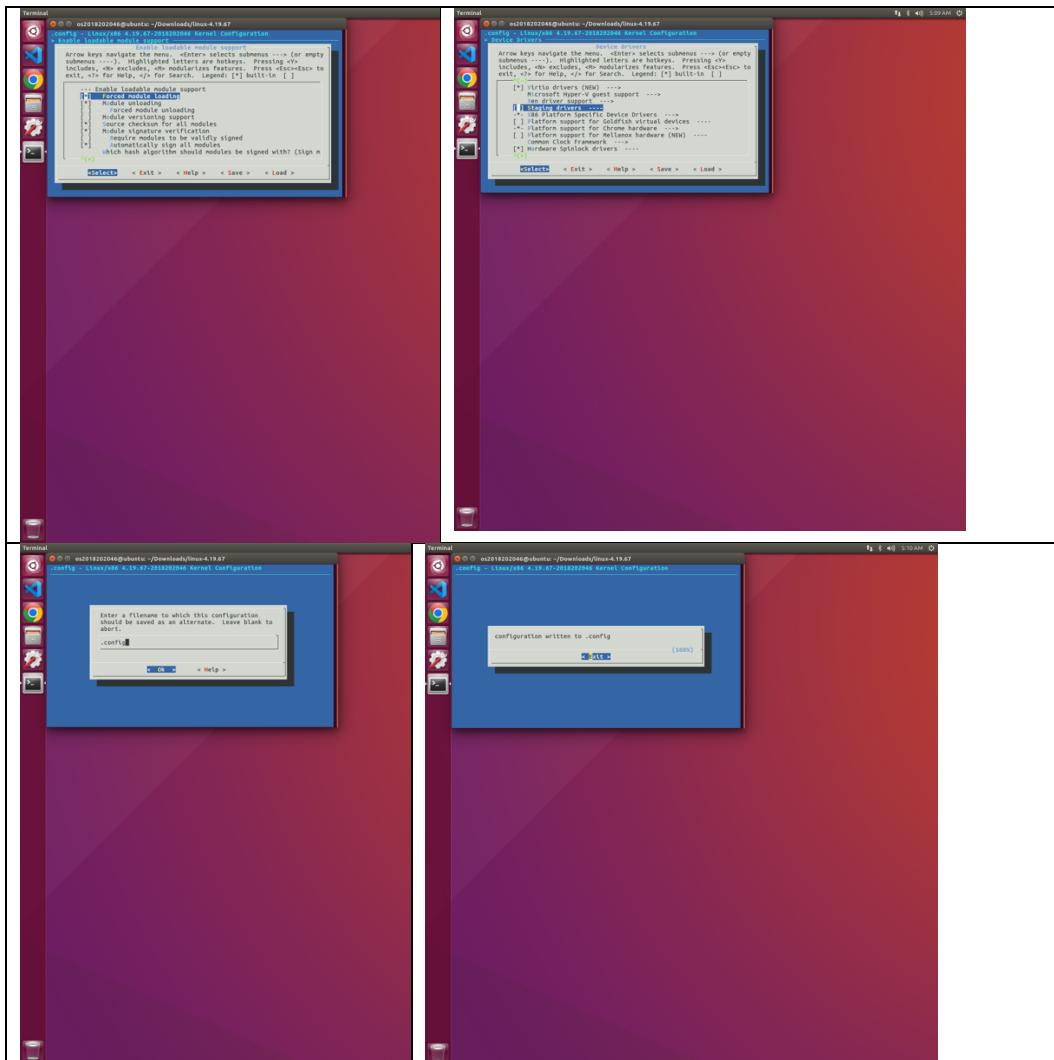
The image shows a single terminal window on an Ubuntu desktop. The user is running the command `sudo apt install build-essential libncurses5-dev libssl-dev libelf-dev` to install the required build tools and libraries for kernel compilation.

```
os2018202046@ubuntu: ~/Downloads/linux-4.19.67$ sudo apt install build-essential libncurses5-dev libssl-dev libelf-dev
```

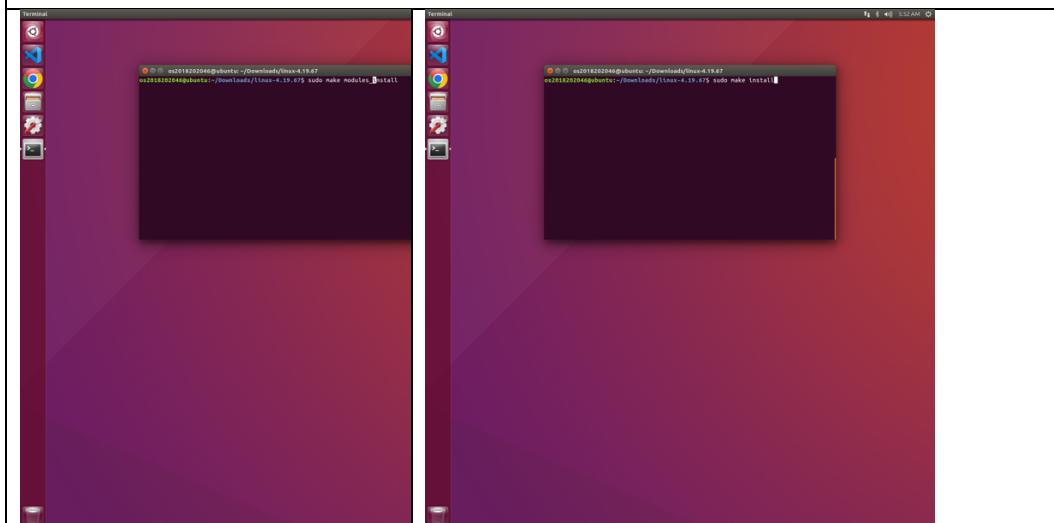
그 후 커널 환경을 설정하기 위해 sudo apt install을 통해 build-essential,

libncurses5-dev, bison, libelf-dev를 설치한다.



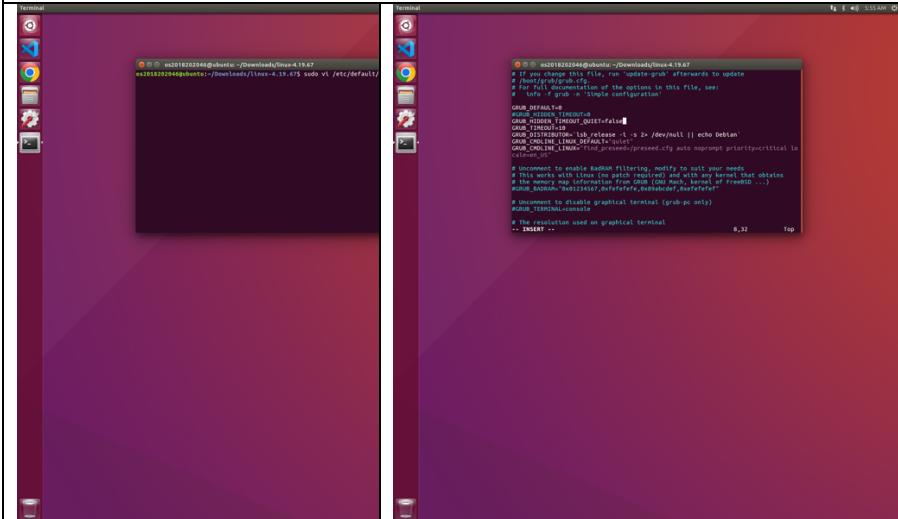


그 다음으로는 Kernel의 환경을 설정한다. sudo make menuconfig를 입력하여 환경 설정으로 들어간다. 그 후 커널 모듈 적재 시 발생할수 있는 문제 해결하고, 컴파일 시 문제가 될 수 있는 모듈을 제거한다. 마지막으로 이를 .config에 저장한 후 환경설정을 완료한다.

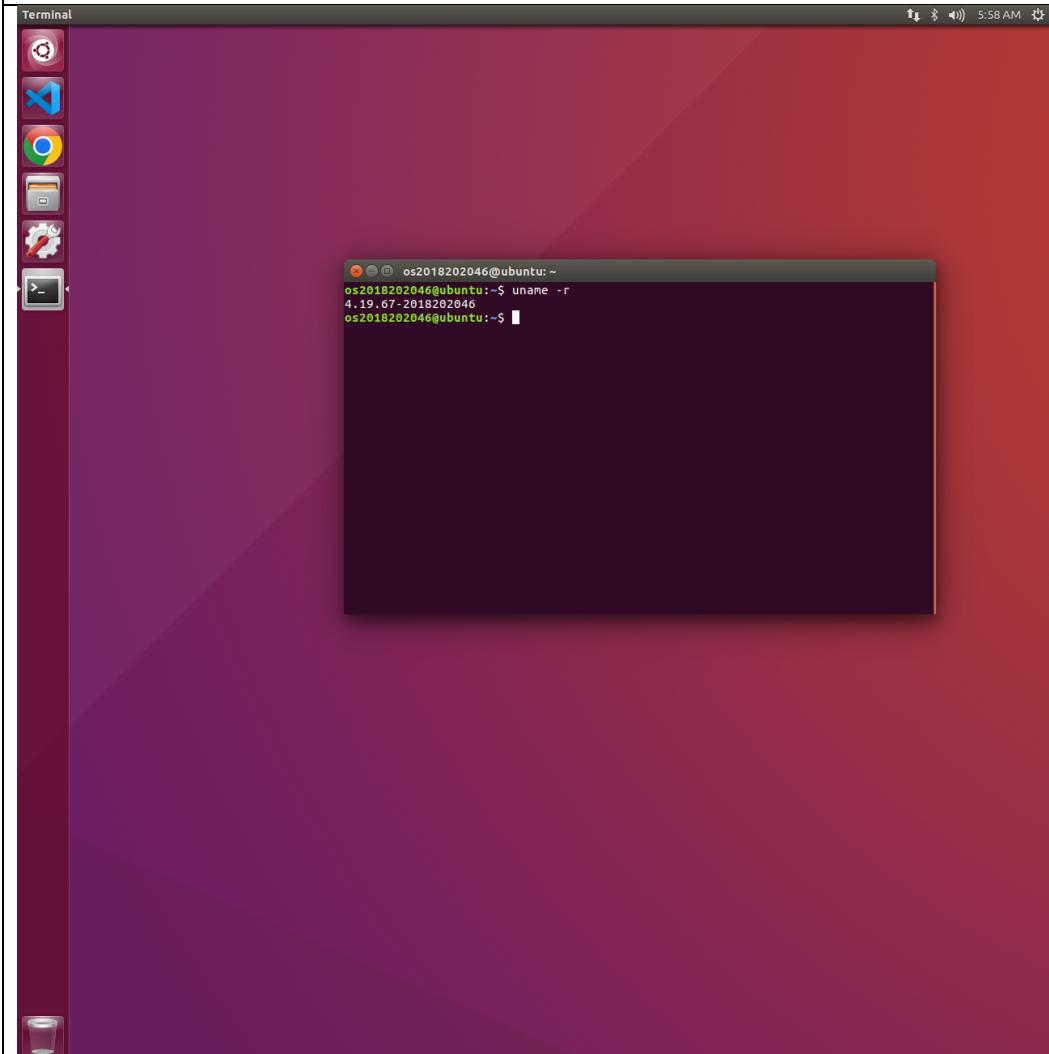


이후 make j4를 통해 4개의 thread를 통해 kernel을 컴파일한 후 make

modules\_install을 통해 모듈을 설치, 그 후 make install을 통해 컴파일된 kernel을 boot Loader에 등록한다.



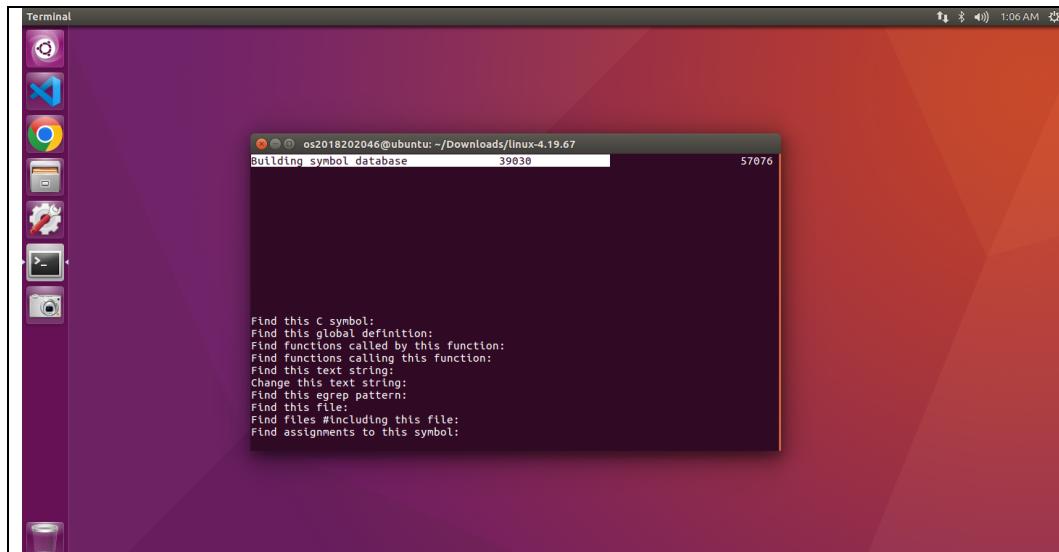
그 후 Grub 설정파일에 들어가 Grub 메뉴 숨김 옵션을 false로 설정하고 Grub 메뉴를 숨기고 기다리는 시간을 0으로 설정한다.



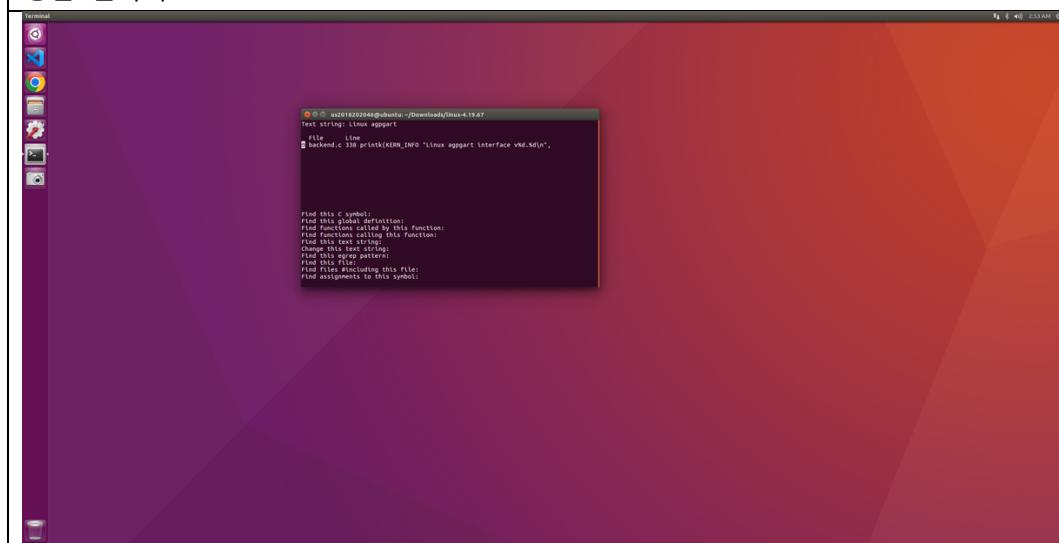
위의 모든 작업을 완료하고 reboot를 이용하여 재부팅을 하여 Grub 부트로더 선택

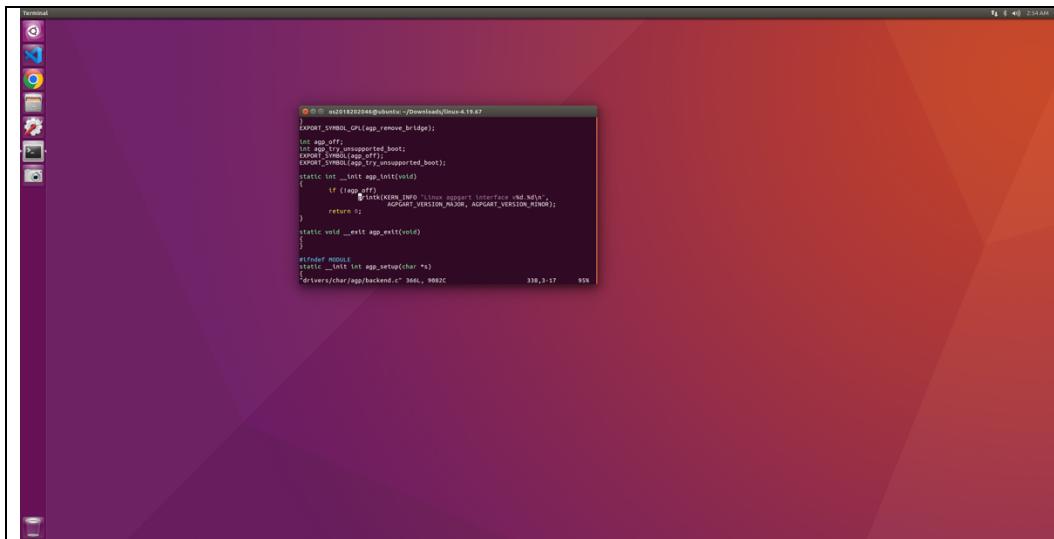
메뉴에서 컴파일한 커널을 선택하였을 때 만들어진 커널로 정상적으로 바뀐 것을 확인할 수 있었다.

### C. Assignment 1-3



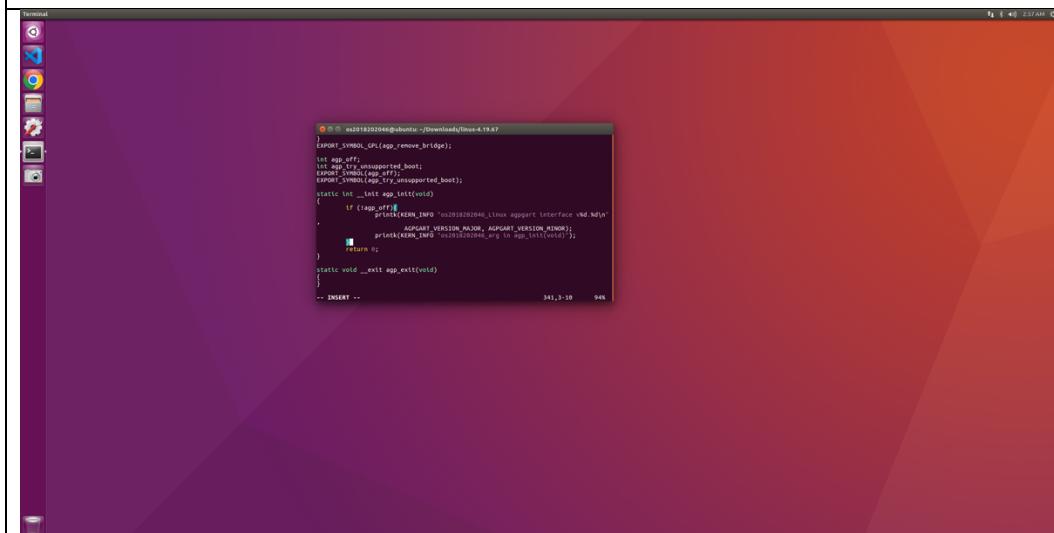
해당 화면은 cscope -R를 통해 cscope의 DB를 생성하고 cscope를 여는 명령을 수행한 결과다.





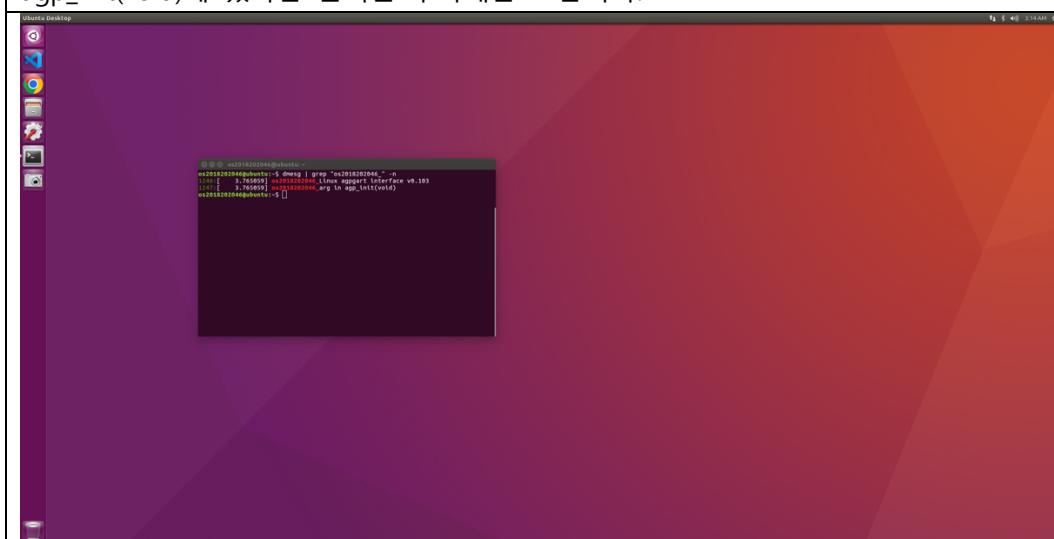
```
EXPORT_SYMBOL_GPL(agp_remove_bridge);
int agp_off;
EXPORT_SYMBOL(agp_off);
EXPORT_SYMBOL(agp_on);
EXPORT_SYMBOL(agp_try_unsupported_boot);
static int __init agp_init(void)
{
    if (!agp_off)
        printk(KERN_INFO "Linux agpgart interface v%d.%d\n",
               AGPGART_VERSION_MAJOR, AGPGART_VERSION_MINOR);
    return 0;
}
static void __exit agp_exit(void)
{
}
#endif
static __init int agp_setup(char *str)
{
    drivers/char/agp/backend.c" 366, 980C           338, 3-17   95K
```

Linux agpgart 출력되는 곳을 탐색하기 위해 Find this text string 항목에 해당 text를 입력하여 파일을 찾았다.



```
EXPORT_SYMBOL_GPL(agp_remove_bridge);
int agp_off;
EXPORT_SYMBOL(agp_off);
EXPORT_SYMBOL(agp_on);
EXPORT_SYMBOL(agp_try_unsupported_boot);
static int __init agp_init(void)
{
    if (!agp_off)
        printk(KERN_INFO "os2018202046_Linux agpgart interface v%d.%d\n",
               AGPGART_VERSION_MAJOR, AGPGART_VERSION_MINOR);
    else
        printk(KERN_INFO "os2018202046_agp in agp_init(void)\n");
    return 0;
}
static void __exit agp_exit(void)
{
}
-- INSERT --           341,3-10   94K
```

해당 함수의 출력을 정해진 조건과 같이 변경하였으며 현재 해당 파일이 agp\_init(void)에 있다는 출력을 추가해준 모습이다.



```
root@os2018202046:~# dmesg | grep "os2018202046"
[    0.775858] os2018202046 Linux agpgart interface v0.10
[    0.775860] os2018202046_agp in agp_init(void)
root@os2018202046:~#
```

위와 같이 파일을 수정한 후 make – make modules\_install – make install – reboot 과정을 통해 module을 컴파일 한 후 reboot를 한 후다. dmesg를 통해 커널 메세지를 출력할 때 grep을 통해 \_os2018202046가 출력되는 부분을 찾을 때 다음과 같이 정상적으로 출력이 나오는 것을 확인할 수 있다. 이를 통해 해당 과제가 정상적으로 수행되었음을 확인하였다.

### 3. Consideration

해당 과제를 통해 Linux의 기본 command에 대해 복습하는 시간을 가질 수 있었다. 기본적인 command와 함께 vim의 사용법과 cscope의 사용법을 알 수 있었다. 또한 커널의 컴파일을 하고 적용하는 과정이 어떻게 진행되는지 공부하고 이를 실제로 구현해볼 수 있는 과제였다. 마지막으로 커널에 message를 출력하는 방법을 배우고, 이를 dmesg 명령을 통해 확인하며, grep 명령을 통해 특정 부분만을 선별하는 방법을 익힐 수 있었다.

### 4. Reference

- 강의자료만을 참고