

운영체제

Assignment 1

Class : A
Professor : 김태석 교수님
Student ID : 2022202065
Name : 박나림

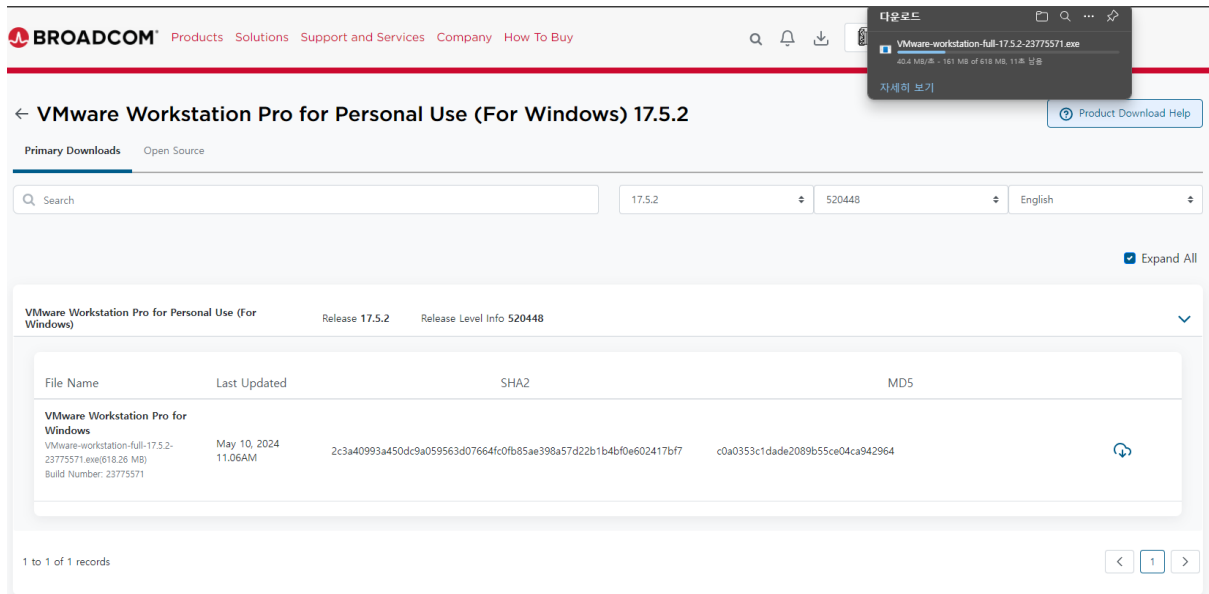
Introduction

운영체제의 기초적인 과정을 수행하는 과제로서, 먼저 가상머신에서 리눅스를 설치한다. 그 후 우분투 내에서 Kernel 을 다운로드하여 컴파일 과정을 확인해보도록 한다. 또한 커널 메시지가 지정된 방식으로 출력되도록 커널 코드를 수정하는 작업을 수행하도록 한다. 이러한 과정을 거치면서 리눅스에서 어떤 방식으로 커널 코드를 작성하고 컴파일이 진행되는지 공부해보는 것을 목표로 한다.

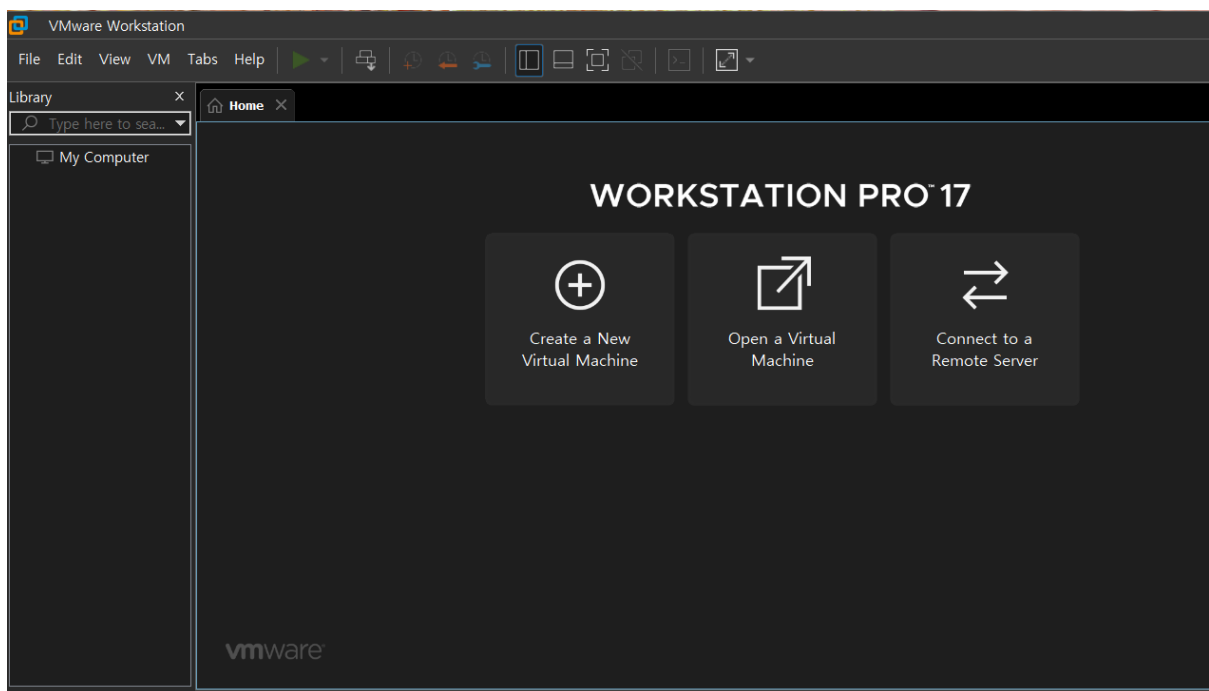
결과화면

Assignment 1-1

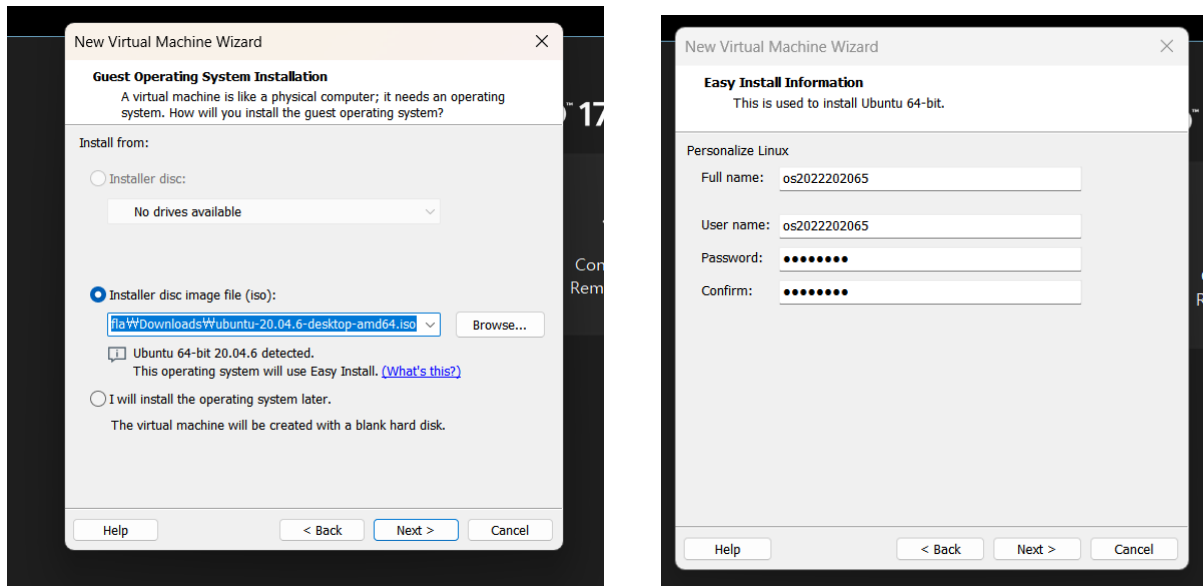
BROADCOM 에 계정을 생성하여 로그인한 뒤, VMWare Workstation Pro 17.5.2 를 설치한다.



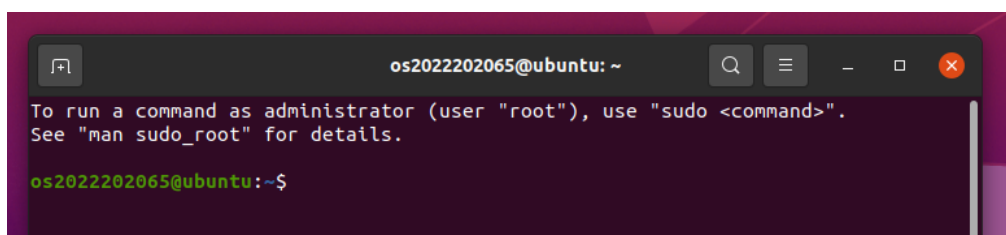
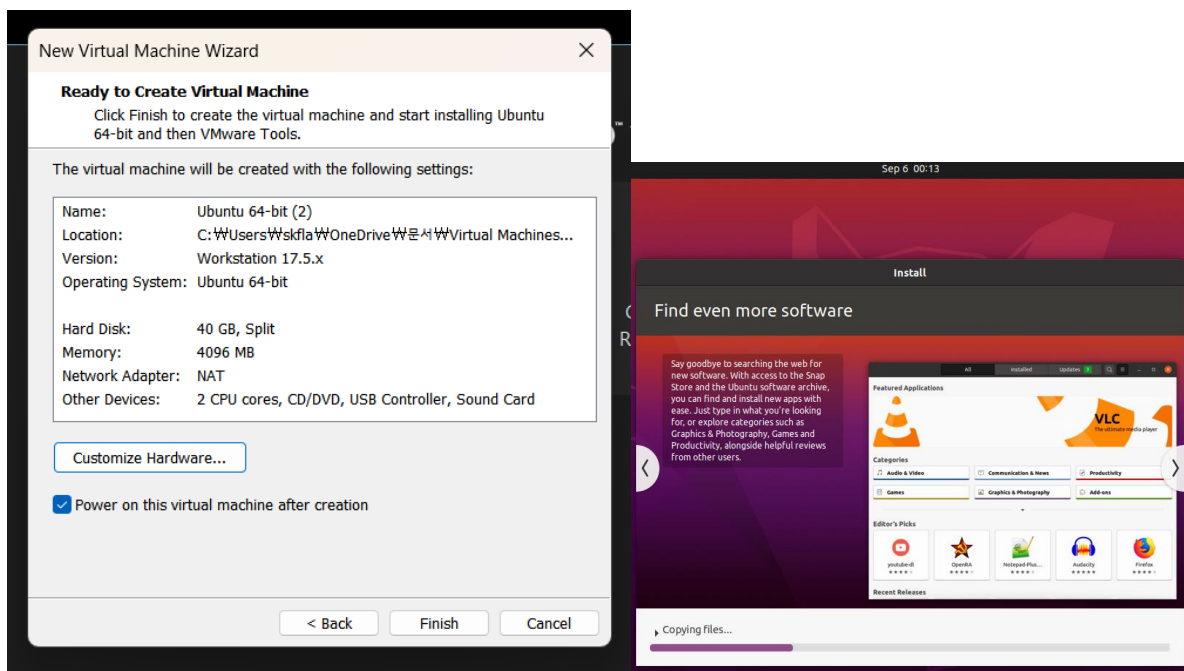
Ubuntu 20.06.04 Desktop 64bit ISO file 도 설치한 뒤, VMWare 를 실행한다.



Ubuntu ISO 파일 위치를 지정해주고 로그인 계정을 생성한다. (os+학번)



가상머신에 할당할 저장장치 크기를 시스템 자체에서 지원하는 수보다 작도록 설정해준 뒤 Finish 를 누르면 자동으로 Ubuntu 가 설치된다.

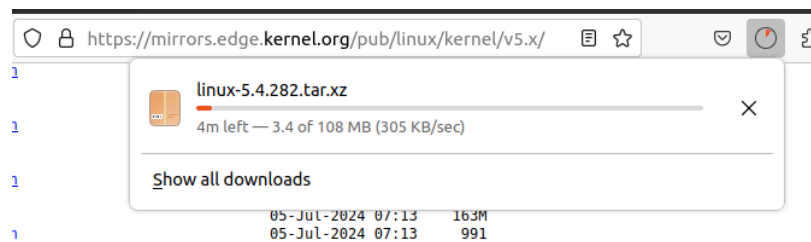


Assignment 1-2

기존 커널 버전은 다음과 같다.

```
os2022202065@ubuntu:~$ uname -a
Linux ubuntu 5.15.0-119-generic #129~20.04.1-Ubuntu SMP Wed Aug 7 13:07:13 UTC 2
024 x86_64 x86_64 x86_64 GNU/Linux
os2022202065@ubuntu:~$ uname -r
5.15.0-119-generic
```

먼저 지정된 커널을 kernel.org 에서 찾아 다운로드 한다. (kernel 5.4.282)



sudo -s 로 루트명령을 가지도록 만들고, 다운로드한 커널 파일을 /usr/src 에 옮긴다.

```
os2022202065@ubuntu:~$ sudo -s
[sudo] password for os2022202065:
root@ubuntu:/home/os2022202065# ls
Desktop Documents Downloads Music Pictures Public Templates Videos
root@ubuntu:/home/os2022202065# cd Downloads/
root@ubuntu:/home/os2022202065/Downloads# ls
linux-5.4.282.tar.xz
root@ubuntu:/home/os2022202065/Downloads# mv linux-5.4.282.tar.xz /usr/src
root@ubuntu:/home/os2022202065/Downloads# cd ..
root@ubuntu:/home/os2022202065# ls
Desktop Documents Downloads Music Pictures Public Templates Videos
root@ubuntu:/home/os2022202065# cd /usr/src
root@ubuntu:/usr/src# ls
linux-5.4.282.tar.xz          linux-hwe-5.15-headers-5.15.0-119
linux-headers-5.15.0-119-generic  linux-hwe-5.15-headers-5.15.0-67
linux-headers-5.15.0-67-generic
```

그후 tar xvf (커널파일) 명령어로 압축을 해제하면 파일이 생성된 것을 볼 수 있다.

```
root@ubuntu:/usr/src# ls
linux-5.4.282
linux-5.4.282.tar.xz
```

필수패키지들도 설치해준다. (git, gcc, 데이터 암호화, 문자 분석기, ELF 파일관리 등)

```
root@ubuntu:/usr/src# sudo apt-get -y install git fakeroot build-essential ncurs
es-dev xz-utils libssl-dev bc flex libelf-dev bison
```

```
root@ubuntu:/usr/src# sudo apt install dwarves zstd
```

```
root@ubuntu:/usr/src# sudo apt install qt5*
```

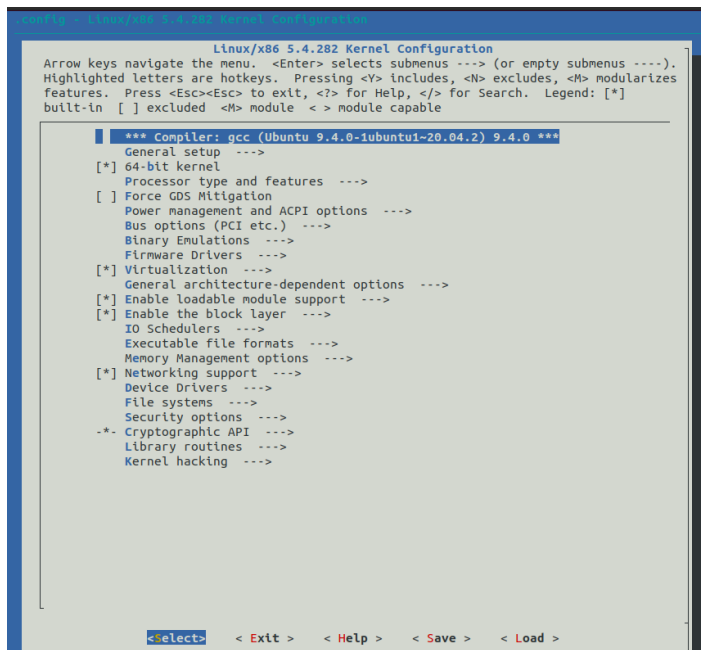
커널 설정

설치가 완료되고 나면 압축을 해제한 커널 파일로 이동한 뒤, cp 명령어로 기존 커널(5.15.0-119-generic)의 config 를 새로 컴파일할 커널 디렉토리에 복사한다.

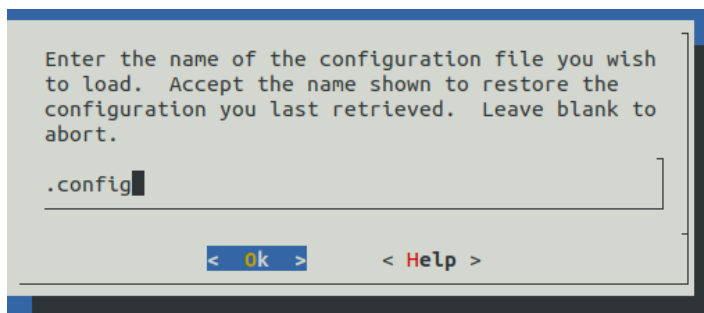
```
root@ubuntu:/usr/src/linux-5.4.282# cp /boot/config-5.15.0-119-generic ./config
```

그 다음, make menuconfig 로 UI 를 통해 모든 설정값을 설정할 수 있다.

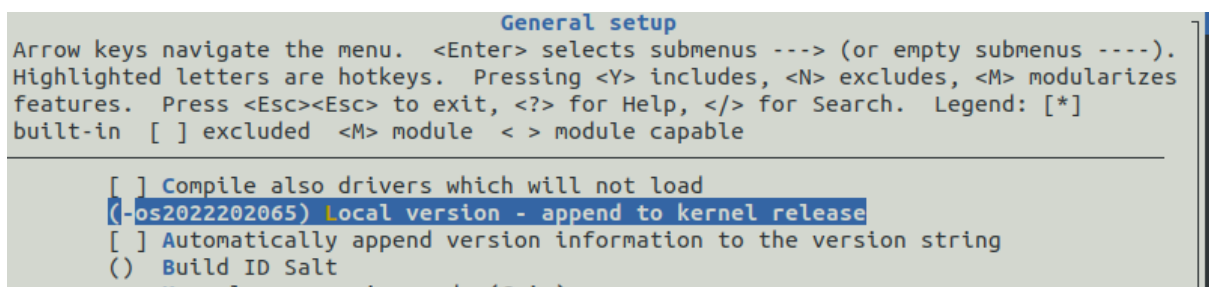
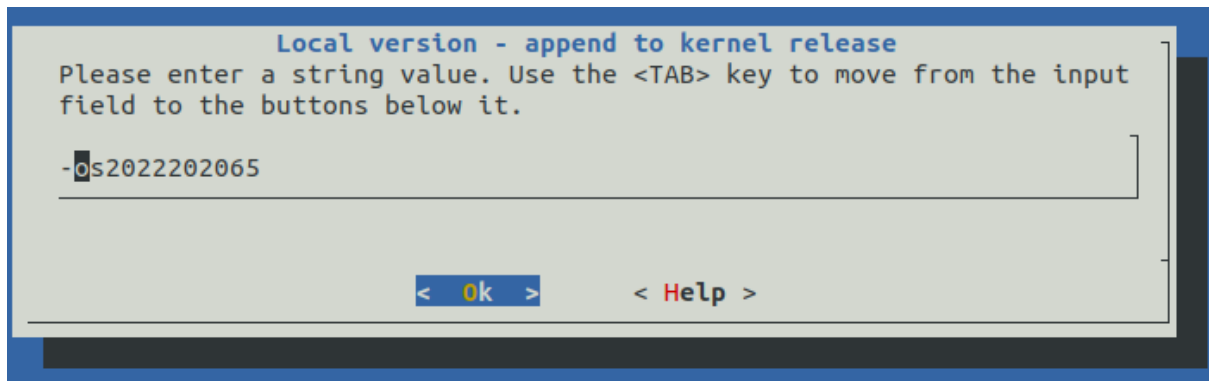
```
root@ubuntu:/usr/src/linux-5.4.282# sudo make menuconfig
```



기존 리눅스 배포판이 사용한 config 설정을 그대로 사용해서 대부분은 변경할 것이 없다. 키보드 키로 Load->ok, Save->ok 를 한다.



uname -r 의 결과가 "5.4.282-os2022202054"로 나오기 위해서는 이름 설정을 바꿔주어야 한다. 첫 화면에서 General Setup->Local version 으로 이동해서 "-학번"을 입력하고 save 한 뒤 exit 로 종료한다.



커널 컴파일

커널을 빌드 하기 전 다음 명령어를 입력한다.

```
root@ubuntu:/usr/src/linux-5.4.282# sudo scripts/config --disable SYSTEM_TRUSTED_KEYS
root@ubuntu:/usr/src/linux-5.4.282# sudo scripts/config --disable SYSTEM_REVOCATION_KEYS
```

커널 디렉토리 안에 Makefile 이 있으므로 make 명령어를 입력하면 바로 커널을 빌드할 수 있다. 이때 병렬 처리로 시간을 줄이기 위해 -j 2 를 입력하여 프로세스 2 개로 진행하도록 만든다.

```
root@ubuntu:/usr/src/linux-5.4.282# make -j 2
HOSTCC scripts/kconfig/conf.o
HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --synconfig Kconfig
*
* Restart config...
*
```

이때 아래와 같은 오류가 나면 sudo vi .config 파일에 들어가서 esc 로 /CONFIG_SYSTEM_...을 검색한 뒤 공백으로 바꾸어준다.

```
make[1]: *** No rule to make target 'y', needed by 'certs/x509_certificate_list'. Stop.
make[1]: *** Waiting for unfinished jobs....
make: *** [Makefile:1750: certs] Error 2
make: *** Waiting for unfinished jobs....
```

```
CONFIG_SYSTEM_TRUSTED_KEYS="y"
```

->

```
CONFIG_SYSTEM_TRUSTED_KEYS=""
CONFIG_SYSTEM_EXTRA_CERTIFICATE=y
CONFIG_SYSTEM_EXTRA_CERTIFICATE_SIZE=4096
CONFIG_SECONDARY_TRUSTED_KEYRING=y
CONFIG_SYSTEM_BLACKLIST_KEYRING=y
CONFIG_SYSTEM_BLACKLIST_HASH_LIST=""
CONFIG_SYSTEM_REVOCATION_LIST=""
```

Module 도 make 한다. 완료된 후 modules.builtin 이 있는지 확인한다.

```
root@ubuntu:/usr/src/linux-5.4.282# make modules
CALL scripts/checksyscalls.sh
CALL scripts/atomic/check-atomics.sh
DESCEND objtool
CHK include/generated/compile.h
CHK kernel/kheaders_data.tar.xz
Building modules, stage 2.
MODPOST 5310 modules
root@ubuntu:/usr/src/linux-5.4.282# ls
arch      crypto    init      lib        modules.builtin  README      System.map  vmlinux-gdb.py
block     Documentation  ipc       LICENSES   modules.builtin.modinfo  samples     tools       vmlinux.o
certs     drivers    Kbuild   MAINTAINERS  modules.order     scripts     usr
COPYING   fs         Kconfig  Makefile    Module.symvers    security    virt
CREDITS   include    kernel   mm          net                sound       vmlinux
```

커널이 빌드된 후 sudo make modules_install 로 컴파일 된 모듈을 /lib/modules 로 이동시킨다.

```
root@ubuntu:/usr/src/linux-5.4.282# sudo make modules_install
```

완료되면 sudo make install 로 컴파일 된 커널을 부트로더에 등록한다.

```
root@ubuntu:/usr/src/linux-5.4.282# sudo make install
sh ./arch/x86/boot/install.sh 5.4.282-os2022202065 arch/x86/boot/bzImage \
    System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 5.4.282-os2022202065 /boot/vmlinuz-5.4.282-os2022202065
update-initramfs: Generating /boot/initrd.img-5.4.282-os2022202065
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 5.4.282-os2022202065 /boot/vmlinuz-5.4.282-os2022202065
run-parts: executing /etc/kernel/postinst.d/update-notifier 5.4.282-os2022202065 /boot/vmlinuz-5.4.282-os2022202065
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 5.4.282-os2022202065 /boot/vmlinuz-5.4.282-os2022202065
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-119-generic
I: /boot/initrd.img is now a symlink to initrd.img-5.4.282-os2022202065
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 5.4.282-os2022202065 /boot/vmlinuz-5.4.282-os2022202065
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-5.15.0-119-generic
Found initrd image: /boot/initrd.img-5.15.0-119-generic
Found linux image: /boot/vmlinuz-5.15.0-67-generic
Found initrd image: /boot/initrd.img-5.15.0-67-generic
Found linux image: /boot/vmlinuz-5.4.282-os2022202065
Found initrd image: /boot/initrd.img-5.4.282-os2022202065
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```

작업 완료 후 재부팅한다.

```
root@ubuntu:/usr/src/linux-5.4.282# sudo reboot
```


재부팅 후 접속해서 `uname -r` 로 확인해보면 기존 버전으로 안 바뀐 상태로 뜨는데, GRUB(부트로더로 켜질 때 가장 먼저 실행되는 프로그램)설정을 따로 해주어야 하기 때문이다. `Sudo vim /etc/default/grub` 으로 파일을 열고 아래처럼 3 가지를 수정해준다.

```
GRUB_TIMEOUT_STYLE=menu
```

```
GRUB_TIMEOUT=10
```

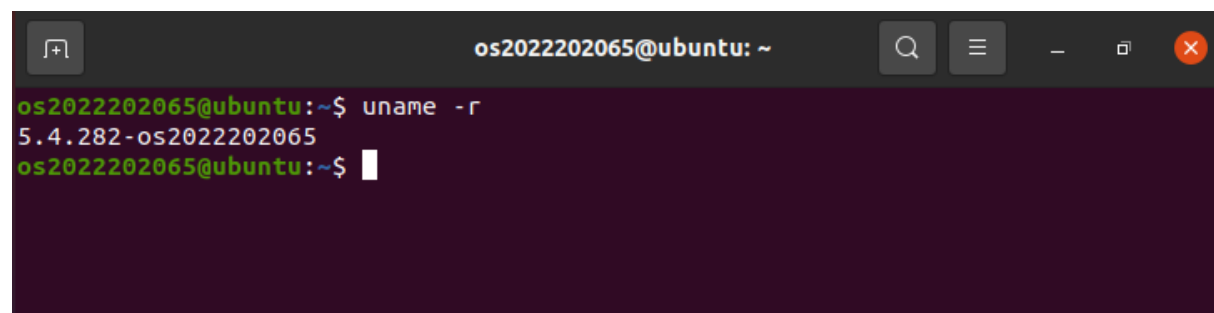
```
GRUB_TERMINAL=console (주석해제)
```

```
GRUB_DEFAULT=0
GRUB_TIMEOUT_STYLE=menu
GRUB_TIMEOUT=10
```

저장 후 다시 재부팅 하면 부트로더 menu 가 10 초간 뜬다. 그때 방향키로 Advanced options for Ubuntu -> 5.4.282-os2022202065 (복구 말고 기본 모드)로 접속한다.

커널 확인

접속 후 `uname -r` 로 확인해보면 새로운 커널이 학번과 함께 출력되는 것을 볼 수 있다.

A terminal window titled 'os2022202065@ubuntu: ~' with search, menu, and window control icons. The terminal shows the command 'uname -r' being executed, resulting in the output '5.4.282-os2022202065'. The prompt is 'os2022202065@ubuntu:~\$' with a cursor.

```
os2022202065@ubuntu:~$ uname -r
5.4.282-os2022202065
os2022202065@ubuntu:~$
```

Assignment 1-3

Linux agpgart interface 가 실행되는 지점에서 커널 메시지를 출력해야 한다. Cscope 에서 Find this text string 검색을 통해 파일의 위치를 알 수 있다. (아래는 코드 수정 후 다시 검색했을 때 캡처화면)

```
Text string: Linux agpgart interface

File      Line
0 backend.c 338 printk(KERN_INFO "os2022202065_Linux agpgart interface
              v%d.%d\n",

Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

처음에 start_kernel() 함수에서 부팅을 진행하면서 do_basic_setup() 함수를 통해 커널 드라이버 초기화 작업을 진행한다. 이때 do_initcalls() 함수가 호출되면서 커널에 등록된 모든 모듈의 초기화 함수를 호출하는데, 여기서 agpgart 모듈의 agp_init() 함수가 실행된다. 이 부분이 Linux agpgart interface 가 실행되는 지점이다.

위에서 찾은 backend.c 파일을 오픈한 뒤 Ctags 를 이용하여 ":tj agp_init"을 검색하면 아래와 같이 두 결과가 뜬다. 여기서 찾고자 하는 함수는 첫번째 이므로 1 을 입력한다.

(수정한 소스코드 path: `/usr/src/linux-5.4.282/drivers/char/agp/backend.c`)

```
# pri kind tag          file
1 FSC f      agp_init    /usr/src/linux-5.4.282/drivers/char/agp/backend.c
              static int __init agp_init(void)
2 F C v      agp_init    /usr/src/linux-5.4.282/drivers/char/agp/backend.c
              module_init(agp_init);
Type number and <Enter> (empty cancels):
```

입력하면 아래와 같이 agpgart 가 실행되는 지점인 agp_init 함수가 나타난다.

```
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;
}
```

이를 지정된 형식대로 다음과 같이 수정한다.

```
static int __init agp_init(void)
{
    if (!agp_off) {
        printk(KERN_INFO "os2022202065_Linux agpgart interface v%d.%d\n",
                AGPGART_VERSION_MAJOR, AGPGART_VERSION_MINOR);
        printk(KERN_INFO "os2022202065_arg in agp_init(void)\n");
    }
    return 0;
}
```

저장한 뒤, module compile 후 reboot 하여 확인한 결과는 다음과 같다.

수정전에 dmesg 를 통해 검색한 결과는 기본 상태로 출력되었는데,

```
os2022202065@ubuntu:~$ dmesg | grep agpgart
[ 24.789039] Linux agpgart interface v0.103
```

수정 후 reboot 하여 다시 dmesg 를 통해 확인하면 아래와 같이 변경된 것을 볼 수 있다.
(1287, 1288)

```
os2022202065@ubuntu:~$ dmesg | grep "os2022202065" -n
1:[ 0.000000] Linux version 5.4.282-os2022202065 (root@ubuntu) (
2:[ 0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-5.4.282-os2
420:[ 0.319347] Kernel command line: BOOT_IMAGE=/boot/vmlinuz-5.
_US quiet
1287:[ 8.211891] os2022202065_Linux agpgart interface v0.103
1288:[ 8.211892] os2022202065_arg in agp_init(void)
1310:[ 8.392855] usb usb1: Manufacturer: Linux 5.4.282-os2022202
1326:[ 8.395649] usb usb2: Manufacturer: Linux 5.4.282-os2022202
os2022202065@ubuntu:~$
```

고찰

Assignment1-2 의 커널 설치 과정에서 여러 시행착오가 있었다. 처음 설치했을 때는 터미널에 에러 문구는 나타나지 않았지만 마지막에 재부팅할 때 검은 화면만 나타나고 부팅이 안되는 상황이 발생하였었다. 다시 삭제하고 두번째 시도를 했을 때는 make 하는 과정에서 에러 문구가 출력되었는데, 용량 때문에 파일들을 다 설치할 수 없다는 에러였다. 이에 다시 재부팅을 해봤으나 첫번째와 마찬가지로 부팅이 되지 않았다. 찾아보니 재부팅이 되지 않을 때는 여러 이유가 있지만 여기서는 에러 내용과 같이 우분투 용량이 가득 차서 부팅이 안되는 것이었다. 그래서 복구 모드로 진입하여 명령어를 치고 확인해보니 커널을 설치한 디스크인 /dev/sda5 가 100%였다. 여기서 clean 으로 사용 안 하는 파일들을 삭제하니 99%로 떨어져서 부팅이 됐었지만 더 진행할 수 없다고 판단하여 다시 삭제하였다. 그 후 처음부터 우분투 disk 를 40GB 에서 50GB 로 늘리고 진행하니 성공적으로 설치될 수 있었다.

Reference

강의자료 참고