

2022학년도 2학기

운영체제

-REPORT-



수업명	운영체제
과제 이름	assignment1
담당 교수님	최상호 교수님
학번	2018204058
이름	김민교

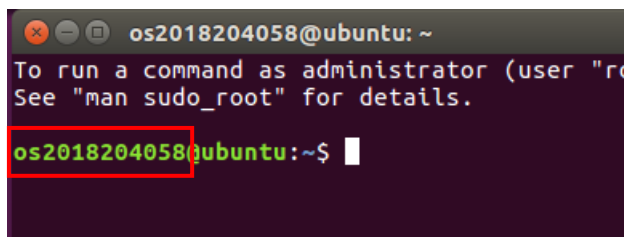
1. Introduction :

가상머신을 통해 리눅스 운영체제를 설치한다. kernel이 start되는 지점을 탐색해 본다. 리눅스 환경에서 명령어를 직접 실행시켜보면서 파일의 생성, 복사, 권한 부여 등의 기능을 익힌다. 이로써 사용자 그래픽 인터페이스의 작동 원리를 유추해본다. 현재 운영체제 커널의 다른 버전을 다운로드하고 컴파일, 설치를 진행해본다. 리눅스 커널이 오픈소스임을 알고 코드를 확인하고 간단한 수정을 해본다. Kernel이 어떤 함수에서 작동되는지 알아 볼 수 있다. 또한 그 함수가 어떤 과정을 거치는지 알아보고 운영체제를 시작하기 위해 많은 것들을 초기화 해야 되는 것을 알 수 있다.

2. Conclusion & Analysis :

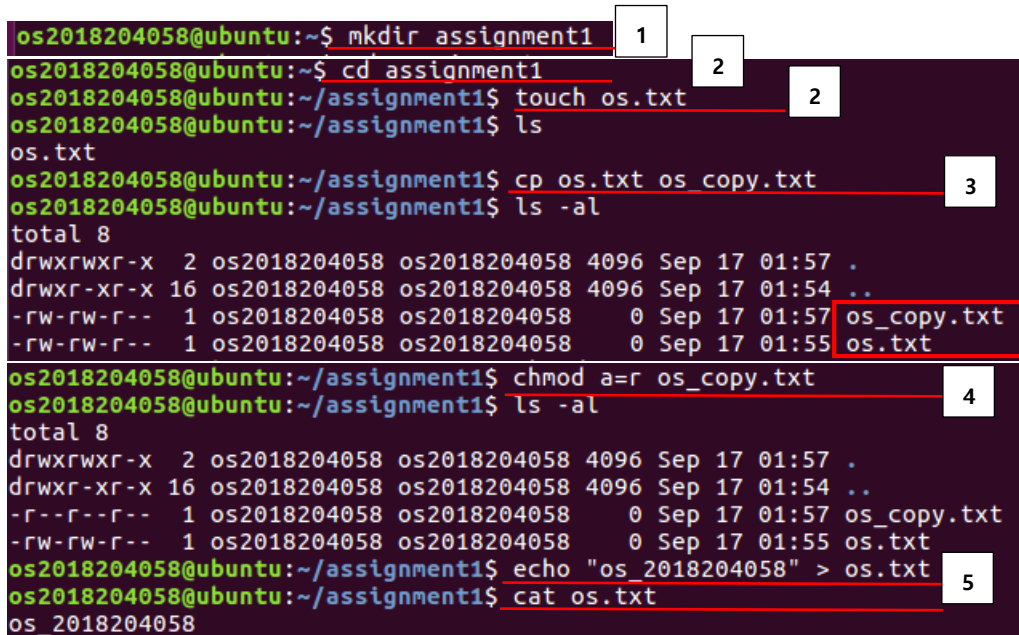
I. Assignment 1-1

[Linux Installation]



```
os2018204058@ubuntu: ~
To run a command as administrator (user "root"), use the sudo command.
See "man sudo_root" for details.
os2018204058@ubuntu:~$
```

[Linux Command]



```
os2018204058@ubuntu:~$ mkdir assignment1
os2018204058@ubuntu:~$ cd assignment1
os2018204058@ubuntu:~/assignment1$ touch os.txt
os2018204058@ubuntu:~/assignment1$ ls
os.txt
os2018204058@ubuntu:~/assignment1$ cp os.txt os_copy.txt
os2018204058@ubuntu:~/assignment1$ ls -al
total 8
drwxrwxr-x 2 os2018204058 os2018204058 4096 Sep 17 01:57 .
drwxr-xr-x 16 os2018204058 os2018204058 4096 Sep 17 01:54 ..
-rw-rw-r-- 1 os2018204058 os2018204058 0 Sep 17 01:57 os_copy.txt
-rw-rw-r-- 1 os2018204058 os2018204058 0 Sep 17 01:55 os.txt
os2018204058@ubuntu:~/assignment1$ chmod a=r os_copy.txt
os2018204058@ubuntu:~/assignment1$ ls -al
total 8
drwxrwxr-x 2 os2018204058 os2018204058 4096 Sep 17 01:57 .
drwxr-xr-x 16 os2018204058 os2018204058 4096 Sep 17 01:54 ..
-r--r--r-- 1 os2018204058 os2018204058 0 Sep 17 01:57 os_copy.txt
-rw-rw-r-- 1 os2018204058 os2018204058 0 Sep 17 01:55 os.txt
os2018204058@ubuntu:~/assignment1$ echo "os_2018204058" > os.txt
os2018204058@ubuntu:~/assignment1$ cat os.txt
os_2018204058
```

II. Assignment 1-2

[Kernel 4.19.67 Compile]

-Download Kernel Source

```
os2018204058@ubuntu:~/assignment1$ cd ..
os2018204058@ubuntu:~$ ls
assignment1  Documents  examples.desktop  Pictures  Templates
Desktop      Downloads  Music             Public    Videos
os2018204058@ubuntu:~$ cd Downloads
os2018204058@ubuntu:~/Downloads$ sudo wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.19.67.tar.xz
--2022-09-17 02:20:39-- 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 48.4MB/s in 2.0s

2022-09-17 02:20:42 (48.4 MB/s) - 'linux-4.19.67.tar.xz' saved [103291756/103291756]
```

/home/os2018204058/Downloads 경로에 커널 4.19.67을 다운받았다. sudo 명령어는 최고 관리자 (root) 권한으로 실행할 수 있다.

```
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/
linux-4.19.67/virt/lib/Kconfig
linux-4.19.67/virt/lib/Makefile
linux-4.19.67/virt/lib/irqbypass.c
os2018204058@ubuntu:~/Downloads$ ls
linux-4.19.67  linux-4.19.67.tar.xz
```

ls 명령어로 디렉토리 내의 파일을 확인했다. Linux-4.19.67.tar.xz파일이 잘 다운 받았다.

```
os2018204058@ubuntu:~/Downloads$ tar -Jxvf linux-4.19.67.tar.xz
```

tar 명령어를 통해 압축을 해제한다. tar 명령어는 여러 개의 파일을 하나의 파일로 묶거나 풀 때 사용하는 명령어다. -jxvf 옵션은 현재 디렉토리에 압축을 푼다는 의미다.

```
os2018204058@ubuntu:~/Downloads$ cd linux-4.19.67/
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ ls
arch      CREDITS    firmware  ipc       lib        mm         scripts   usr
block     crypto     fs         Kbuild    LICENSES   net        security  virt
certs     Documentation  include   Kconfig   MAINTAINERS  README    sound
COPYING   drivers    init       kernel    Makefile    samples   tools
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ vi Makefile
```

압축을 해제한 후, linux-4.19.67로 경로를 바꾼다. 경로 내의 파일을 살펴보면 Makefile이 있다. vi Makefile 명령어를 통해 파일 속 EXTRAVERSION을 수정해야한다.

```
# SPDX-License-Identifier: GPL-2.0
VERSION = 4
PATCHLEVEL = 19
SUBLEVEL = 67
EXTRAVERSION = -2018204058
NAME = "People's Front"
```

수정을 하고 esc를 누르고 :wq로 빠져나온다.

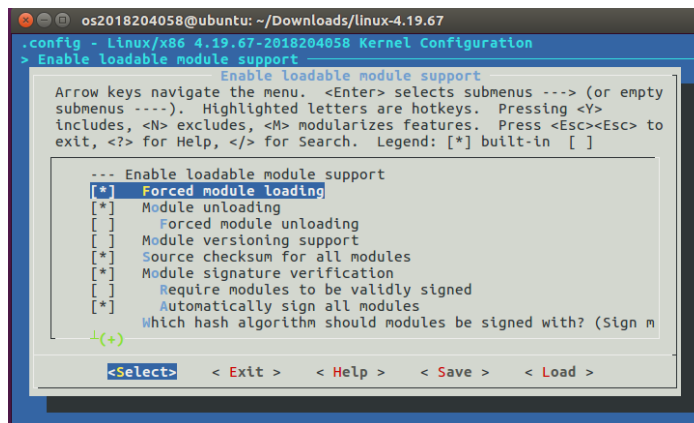
-Kernel 환경 설정

```
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ sudo apt install build-essential
libncurses5-dev bison flex libssl-dev libelf-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

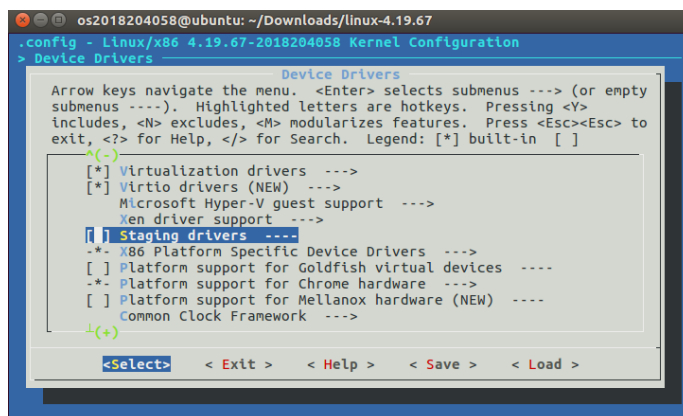
필요한 파일을 sudo 명령어를 통해 다운받는다.

```
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ sudo make menuconfig
```

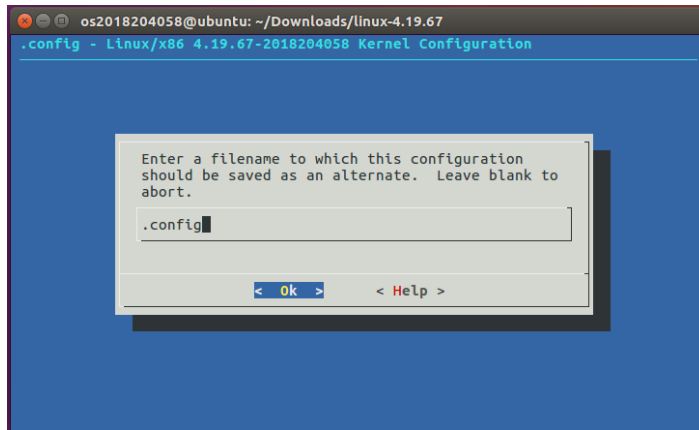
make menuconfig는 리눅스 커널을 컴파일 할 때 필요한 설정을 저장하는 파일을 만드는 명령어이다. make config 명령어도 있다. make menuconfig는 그래픽 모드를 지원한다.



Forced module loading에 스페이스 입력하여 [*]로 변환했다.



Staging drivers에 스페이스를 입력하여 []로 변환했다.



.config 파일로 저장하고 save 했다.

```
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
os2018204058@ubuntu:~/Downloads/linux-4.19.67$
```

그래픽 모드 종료 후 모습

-Kernel Compile

```
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ make -j8
```

make -jn 명령어로 컴파일한다. n은 컴파일을 나누어 수행할 thread 수를 결정한다. 보통 가상머신에 할당한 CPU core수 수의 1.5배 ~ 2배가 적합하다고 한다. 가상머신에 코어를 4개 할당했기 때문에 8로 설정했다.

```
LD [M] sound/usb/line6/snd-usb-line6.ko
LD [M] sound/usb/line6/snd-usb-pod.ko
LD [M] sound/usb/line6/snd-usb-podhd.ko
LD [M] sound/usb/line6/snd-usb-toneport.ko
LD [M] sound/usb/line6/snd-usb-variax.ko
LD [M] sound/usb/misc/snd-ua101.ko
LD [M] sound/usb/snd-usbmidi-lib.ko
LD [M] sound/usb/snd-usb-audio.ko
LD [M] sound/usb/usx2y/snd-usb-us122l.ko
LD [M] sound/usb/usx2y/snd-usb-usx2y.ko
LD [M] sound/x86/snd-hdmi-lpe-audio.ko
LD [M] virt/lib/irqbypass.ko
os2018204058@ubuntu:~/Downloads/linux-4.19.67$
```

30분 정도 소요된 후 컴파일이 완료되었다.

-Module install

```
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ sudo make modules_install
```

```

INSTALL sound/usb/hiface/snd-usb-hiface.ko
INSTALL sound/usb/line6/snd-usb-line6.ko
INSTALL sound/usb/line6/snd-usb-pod.ko
INSTALL sound/usb/line6/snd-usb-podhd.ko
INSTALL sound/usb/line6/snd-usb-toneport.ko
INSTALL sound/usb/line6/snd-usb-variak.ko
INSTALL sound/usb/misc/snd-ua101.ko
INSTALL sound/usb/snd-usb-audio.ko
INSTALL sound/usb/snd-usbmidi-lib.ko
INSTALL sound/usb/usx2y/snd-usb-us122l.ko
INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL sound/x86/snd-hdmi-lpe-audio.ko
INSTALL virt/lib/irqbypass.ko
DEPMOD 4.19.67-2018204058
os2018204058@ubuntu:~/Downloads/linux-4.19.67$

```

-Compile된 Kernel을 Boot Loader에 등록

```

os2018204058@ubuntu:~/Downloads/linux-4.19.67$ sudo make install

```

```

os2018204058@ubuntu:~/Downloads/linux-4.19.67
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 4.19.67-2018204058
/boot/vmlinuz-4.19.67-2018204058
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.19.67-2018204058 /
boot/vmlinuz-4.19.67-2018204058
update-initramfs: Generating /boot/initrd.img-4.19.67-2018204058
run-parts: executing /etc/kernel/postinst.d/pm-utils 4.19.67-2018204058 /boot/vm
linuz-4.19.67-2018204058
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 4.19.67-20182040
58 /boot/vmlinuz-4.19.67-2018204058
run-parts: executing /etc/kernel/postinst.d/update-notifier 4.19.67-2018204058 /
boot/vmlinuz-4.19.67-2018204058
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 4.19.67-2018204058 /b
oot/vmlinuz-4.19.67-2018204058
Generating grub configuration file ...
Warning: Setting GRUB_TIMEOUT to a non-zero value when GRUB_HIDDEN_TIMEOUT is se
t is no longer supported.
Found linux image: /boot/vmlinuz-4.19.67-2018204058
Found initrd image: /boot/initrd.img-4.19.67-2018204058
Found linux image: /boot/vmlinuz-4.15.0-29-generic
Found initrd image: /boot/initrd.img-4.15.0-29-generic
Found mentest86+ image: /boot/mentest86+.elf
Found mentest86+ image: /boot/mentest86+.bin
done
os2018204058@ubuntu:~/Downloads/linux-4.19.67$

```

-Grub 설정 파일 수정

```

os2018204058@ubuntu:~/Downloads/linux-4.19.67$ sudo vi /etc/default/grub

```

```

os2018204058@ubuntu: ~/Downloads/linux-4.19.67
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
# info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=false
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
GRUB_CMDLINE_LINUX="find_preseed=/preseed.cfg auto noprompt priority=critical lo
cale=en_US"

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
-- INSERT --
8,32 Top

```

GRUB_HIDDEN_TIMEOUT=0을 주석처리하고, GRUB_HIDDEN_TIMEOUT_QUIET을 true에서 false로 설정했다

-현재 kernel 버전 확인

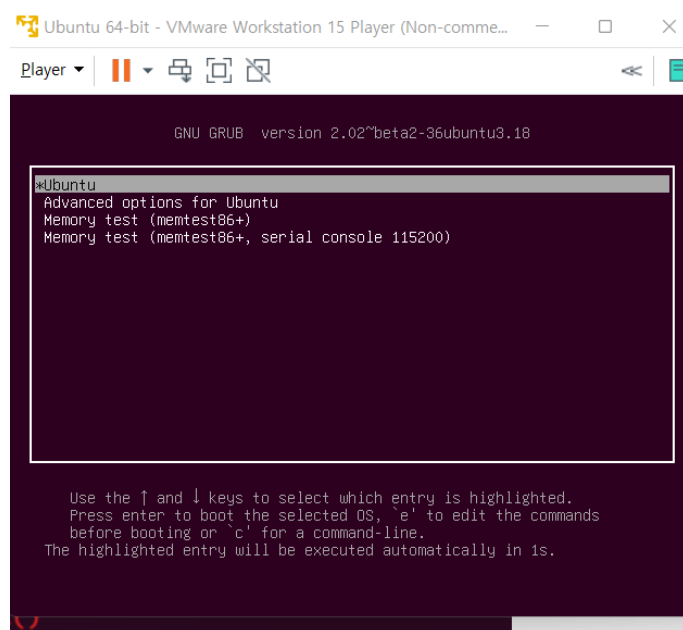
```
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ uname -r
4.15.0-29-generic
```

reboot하기 전, 변화를 보기 위해 현재 커널의 버전을 확인해 보았다. 4.15.0-29-generic이었다. uname 명령어는 현재 시스템 정보를 출력한다. -r 옵션을 주면 kernel release를 출력한다.

-재부팅

```
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ reboot
```

-Grub 부트로더 선택 메뉴에서 컴파일한 커널 선택



-4.19.67 커널로 재부팅 후 버전 확인

```
os2018204058@ubuntu:~$ uname -r
4.19.67-2018204058
```

III. Assignment 1-3

-해결 과정

```
os2018204058@ubuntu:~$ dmesg | grep v0.103
[ 5.667908] Linux agpgart interface v0.103
```

dmesg 명령어를 통해 v0.103 값을 찾아보았다. "Linux agpgart interface v0.103"이 출력되었다. 분명 코드 어딘가 printk("Linux agpgart interface v0.103"); 이라고 적혀있을

것이고, 이 코드가 적힌 곳이 Linux agp...가 실행되는 지점이라고 판단했다.

```
os2018204058@ubuntu: ~/Downloads/linux-4.19.67
Cscope version 15.8b

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

Find this text string에 agpgart를 키워드로 검색을 했다.

```
os2018204058@ubuntu: ~/Downloads/linux-4.19.67
Text string: agpgart

File      Line
0 core_irongate.c 257 alpha_agpgart_size = 0;
1 core_irongate.c 305 #include <linux/agpgart.h>
2 core_irongate.c 324 if (!alpha_agpgart_size)
3 core_irongate.c 338 gart_bus_addr + alpha_agpgart_size)
4 core_marvel.c   906 if (!alpha_agpgart_size)
5 core_marvel.c   913 aper->pg_count = alpha_agpgart_size / PAGE_SI
6 core_marvel.c   962 * The agpgart_be code has not programmed the
7 core_titan.c    593 if (!alpha_agpgart_size)

* Lines 1-9 of 58, 50 more - press the space bar to display more *
```

약 58개의 결과가 도출되었고, 그리 많지 않다고 판단하여 58개중에서 찾기로 했다.

```
os2018204058@ubuntu: ~/Downloads/linux-4.19.67
Text string: agpgart

File      Line
0 agp.h      34 #define PFX "agpgart: "
1 ali-agp.c  399 .name = "agpgart-ali",
2 amd-k7-agp.c 543 .name = "agpgart-amdk7",
3 amd64-agp.c 741 .name = "agpgart-amd64",
4 ati-agp.c   559 .name = "agpgart-ati",
5 backend.c   38 #include <linux/agpgart.h>
6 backend.c  338 printk(KERN_INFO "os2018204058_Linux agpgart interface v%d.%d\n",
7 compat_ioctl.c 32 #include <linux/agpgart.h>

* Lines 23-31 of 58, 28 more - press the space bar to display more *
```

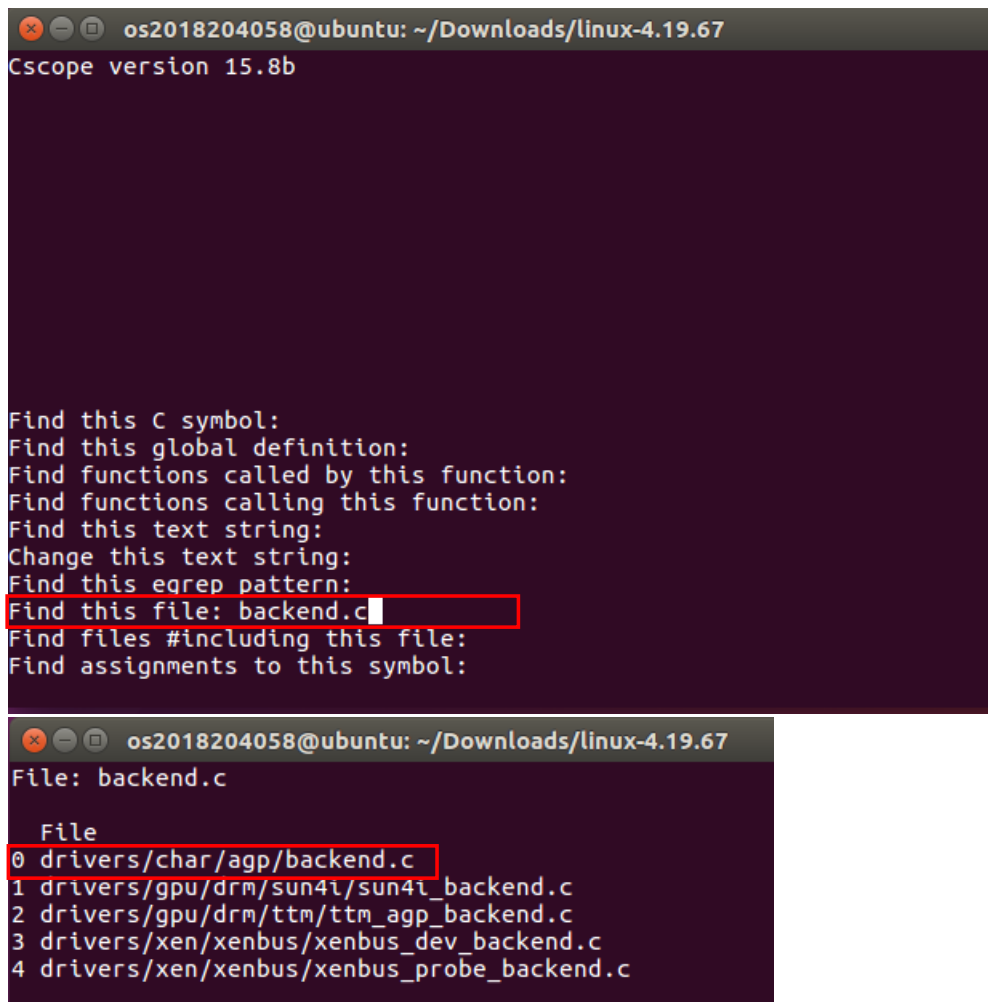
backend.c파일에서 printk(KERN_INFO "Linux agpgart interface v%d.%d\n")를 발견했

다. (제가 처음에 원본 코드를 캡처 못해서 수정 후의 코드로 캡처했습니다.)

```
static int __init agp_init(void)
{
    if (!agp_off)
        printk(KERN_INFO "os2018204058_Linux agpgart interface v%d.%d\n",
                AGPGART_VERSION_MAJOR, AGPGART_VERSION_MINOR);
        printk(KERN_INFO "os2018204058_arg in agp_init(void)\n");
    return 0;
}
```

검색 결과 6번에서 엔터를 치고 확인하니 agp_init(void) 함수에서 Linux agpgart interface가 출력된다는 것을 알았다. 이 지점이 Linux agp... 가실행되는 지점이었고, agp_init 함수가 Linux agp...을 실행시키는 함수인 것 같았다. 그래서 위 사진처럼 출력되도록 수정했다.

(제가 처음에 원본 코드를 캡처 못해서 수정 후의 코드로 캡처했습니다.)



```
os2018204058@ubuntu: ~/Downloads/linux-4.19.67
Cscope version 15.8b

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: backend.c
Find files #including this file:
Find assignments to this symbol:

os2018204058@ubuntu: ~/Downloads/linux-4.19.67
File: backend.c

File
0 drivers/char/agp/backend.c
1 drivers/gpu/drm/sun4i/sun4i_backend.c
2 drivers/gpu/drm/ttm/ttm_agp_backend.c
3 drivers/xen/xenbus/xenbus_dev_backend.c
4 drivers/xen/xenbus/xenbus_probe_backend.c
```

backend.c의 위치를 찾기 위해 검색했다.

backend.c의 경로는 drivers/char/agp/backend.c 다.

커널 코드를 수정한 후, 다시 커널 컴파일을 진행했다

```
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ make
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ sudo make modules_install
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ sudo make install
os2018204058@ubuntu:~/Downloads/linux-4.19.67$ reboot

os2018204058@ubuntu:~$ dmesg | grep "os2018204058" -n
2:[    0.000000] Linux version 4.19.67-2018204058 (os2018204058@ubuntu) (gcc ver
sion 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.10)) #3 SMP Sun Sep 18 01:22:55
PDT 2022
1247:[    5.691621] os2018204058_Linux agpgart interface v0.103
1248:[    5.691622] os2018204058_arg in agp_init(void)
```

reboot 후, dmesg | grep을 통해 "os2018204058" 을 출력했다.

3. 고찰

[assignment 1-1]

과제를 수행하면서 shell prompt에서 명령어를 입력했다. 리눅스 환경에서 파일을 복사하거나 생성하거나 출력하는 명령어를 직접 작성했다. 윈도우 운영체제에서 필자가 마우스 클릭을 통한 파일 생성, 복사 등이 사실은 이런 명령어를 통해 만들어 질 것이라고 예상되었다. 사용자 인터페이스가 있어서 컴퓨터의 대중화가 더욱 잘 되었다고 느꼈다.

[assignment 1-2]

운영체제에는 분명 커널이 포함되어 있을텐데 왜 또 커널을 컴파일 해야하는지 이해가 가지 않았다. 그러나 과제를 수행하면서 커널에도 버전이 있다는 것을 알았다. 커널을 다운받고, 커널 코드에서 단순한 코드를 추가했다. 리눅스는 오픈 소스이다 보니 커널 코드를 전부 공개하는데, 이런 값진 코드를 볼 수 있어서 뜻깊었다.

[assignment 1-3]

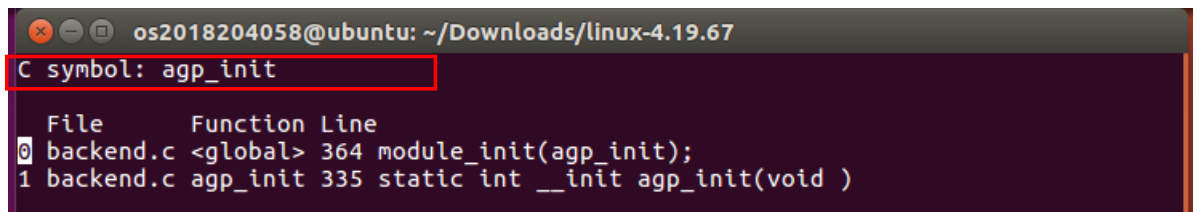
Linux agp...가 실행되는 지점을 찾기 위해 관련 reference를 탐색해보았다. 그러던 중 start_kernel() 함수의 역할을 알게 되었다. start_kernel() 함수의 시작과 함께 커널이 시작한다. start_kernel() 함수는 /init/main.c에 존재한다. 파일을 들여다보면 함수에서 또 다른 함수들을 많이 호출한다. 이들 함수는 운영체제를 시작하기 위해 필요한 것들을 초기화하는 것이다. Reference에 따르면 start_kernel() 함수의 개략적인 동작 순서는 1. Architecture에 특수한(혹은 의존적인) 설정 2. Paging 시스템에 대한 초기화 3. Exception(혹은 trap)에 대한 초기화 4. Interrupt에 대한 초기화 5. Scheduler의 초기화 6. Timer의 초기화 7. Console의 초기화 8. Module들의 초기화 9. Cache와 Buffer의 초기화 10. 메모리 시스템의 초기화 11. 파일 시스템의 초기화 12. ini 시스템 프로세스의 생성 이라고 한다.

또한 agpgart는 AGP 하드웨어를 최대한 활용하는 드라이버라고 한다. 그래픽과 관련이 있다. AGP란 Accelerated Graphics Port의 줄임말로써 3차원 그래픽 표현을 빠르게 구현할 수 있게 해주는 버스 규격이라고 한다. backend.c파일이 왜 drivers 폴더에 있는지 이해가 되었다.

과제를 수행하기 위해 agp가 실행되는 지점에 관련하여 구글링을 많이했다. 그러던 중 dmesg를 정리한 기록을 보게 되었다. 거기서 agp 관련한 msg도 찾았다.

```
(45) -> agp_init()/do_basic_setup()/init()/rest_init()/start_kernel()
Linux agpgart interface v0.99 (c) Jeff Hartmann
agpgart: Maximum main memory to use for agp memory: 439M
agpgart: Detected Via Apollo Pro KT133 chipset
agpgart: AGP aperture is 128M @ 0xe0000000
[drm] AGP 0.99 on VIA Apollo KT133 @ 0xe0000000 128MB
[drm] Initialized mga 3.0.2 20010321 on minor 0
```

agp_init()이라는 함수를 정확하게 알려주고 있었다. 실제로 cscope -R 명령어를 통해 agp_init을 찾으면 바로 접근할 수 있다.



```
os2018204058@ubuntu: ~/Downloads/linux-4.19.67
C symbol: agp_init

File      Function Line
0 backend.c <global> 364 module_init(agp_init);
1 backend.c agp_init 335 static int __init agp_init(void )
```

dmesg 명령어는 시스템 부팅 메시지를 확인하는 명령어다.

grep 명령어는 특정 파일에서 문자열이나 정규 표현식을 포함한 행을 출력해주는 명령어다.

4. Reference

<https://rhrhth23.tistory.com/21> => echo 명령어를 통해 os.txt에 학번을 적는 방법을 습득.

<https://yeslab.tistory.com/41> => menu config 설명

<https://recipes4dev.tistory.com/146#310->

bzip2%EB%A1%9C-%EC%95%95%EC%B6%95%EB%90%9C-

tar-%EC%95%84%EC%B9%B4%EC%9D%B4%EB%B8%8C%EB%A5%BC-%ED%98%84%EC%9

E%AC-%EB%94%94%EB%A0%89%ED%86%A0%EB%A6%AC%EC%97%90-%ED%92%80%EA

%B8%B0 => tar 명령어

<https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=dog9230&logNo=26104267> => start_kernel

<https://forums.gentoo.org/viewtopic-t-117877-start-0.html> => agpgart