

# Assignment 1

수업 명:	운영체제
담당 교수님:	최상호 교수
학번:	2018202074
이름:	김상우
강의시간:	금 1,2

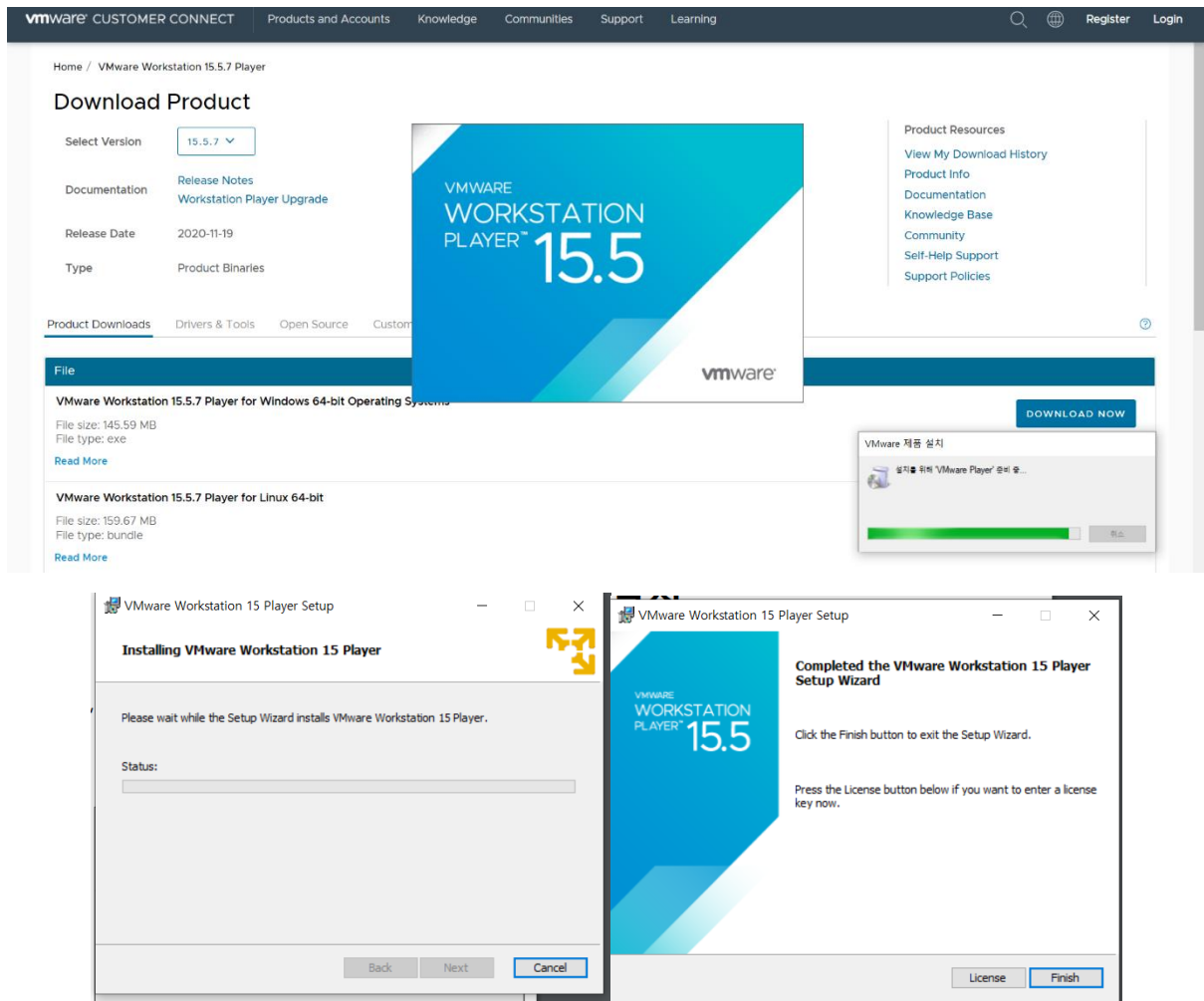
## Introduction

해당 과제는 총 3개의 파트로 구성되어 있습니다. 우선 본격적인 진행에 앞서 리눅스 명령어에 대해 복습을 진행하고, 수업을 위한 기본 환경 구성을 위해 Assignment 1-1이 진행됩니다. 이를 통해 구현된 우분투 환경에서 Assignment 1-2를 통해 kernel을 다운로드 받으며 이를 바탕으로 Assignment 1-3을 진행하며 Linux Kernel Message를 출력해보고 이를 통해 기본적인 구조 이해 및 환경의 확인을 진행합니다. 이 과정에서 cscope, ctags를 활용하며 추가적인 정보를 배울 수 있을 것으로 기대됩니다.

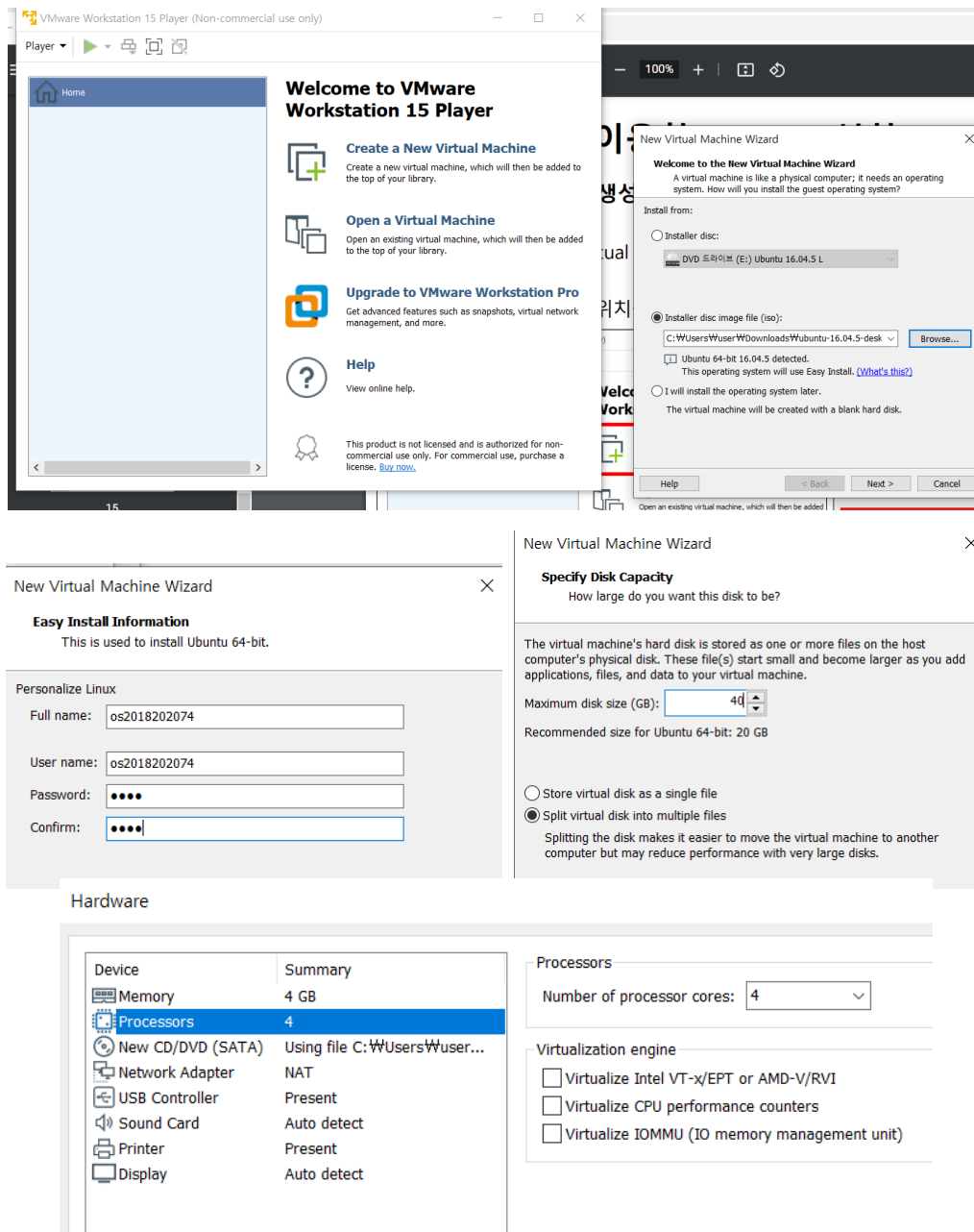
## Conclusion & Analysis

### Assignment1-1-Linux Installation

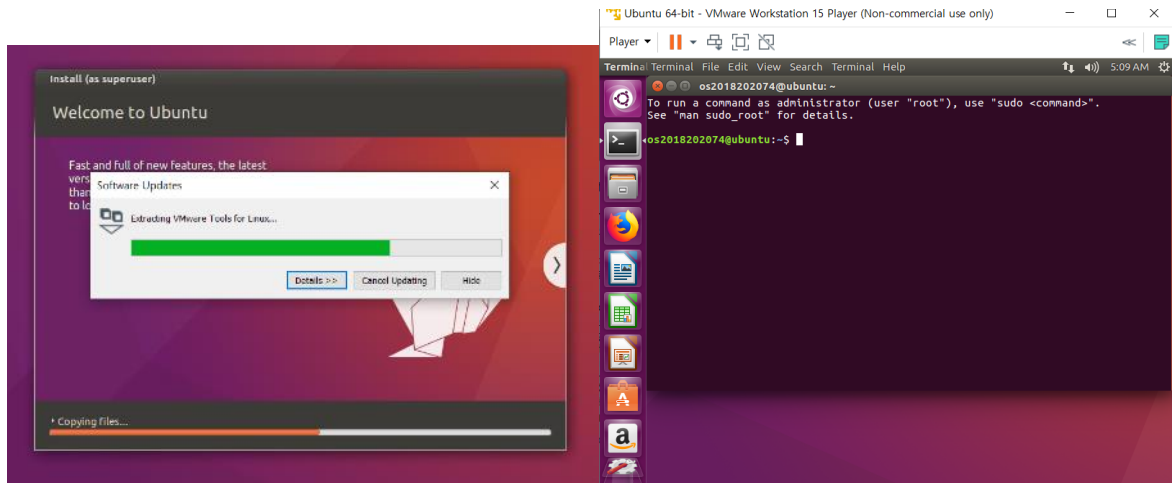
설치를 위해 VM ware를 설치해주었다.



VM Ware가 정상적으로 깔린 것을 확인 후 virtual machine으로서 ubuntu를 실행했다.



운영체제 수업을 위해 계정 ID를 os학번인 os2018202074이며 40GB와 4Core가 할당된 ubuntu를 생성했다.



위 사진의 os2018202074@ubuntu라는 단어를 통해 계정 ID가 알맞게 적용된 것을 확인할 수 있다. 이를 바탕으로 Linux command에 대한 과제를 진행했다.

### Assignment1-1-Linux command

```

os2018202074@ubuntu: ~/assignment1
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

os2018202074@ubuntu:~$ mkdir assignment1
os2018202074@ubuntu:~$ cd assignment1
os2018202074@ubuntu:~/assignment1$ touch os.txt
os2018202074@ubuntu:~/assignment1$ ls
os.txt
os2018202074@ubuntu:~/assignment1$ cp os.txt os_copy.txt
os2018202074@ubuntu:~/assignment1$ ls
os_copy.txt  os.txt
os2018202074@ubuntu:~/assignment1$ ls -al
total 8
drwxrwxr-x  2 os2018202074 os2018202074 4096 Sep  4 00:57 .
drwxr-xr-x 16 os2018202074 os2018202074 4096 Sep  4 00:56 ..
-rw-rw-r--  1 os2018202074 os2018202074    0 Sep  4 00:57 os_copy.txt
-rw-rw-r--  1 os2018202074 os2018202074    0 Sep  4 00:57 os.txt
os2018202074@ubuntu:~/assignment1$ chmod 444 os_copy.txt
os2018202074@ubuntu:~/assignment1$ ls -al
total 8
drwxrwxr-x  2 os2018202074 os2018202074 4096 Sep  4 00:57 .
drwxr-xr-x 16 os2018202074 os2018202074 4096 Sep  4 00:56 ..
-r--r--r--  1 os2018202074 os2018202074    0 Sep  4 00:57 os_copy.txt
-rw-rw-r--  1 os2018202074 os2018202074    0 Sep  4 00:57 os.txt
  
```

1. Assignment1 명의 디렉터리 생성 : `mkdir assignment1`
2. Assignment1 디렉터리 이동 후 : `cd assignment1`  
"os.txt"명의 빈 파일 생성 : `touch os.txt`
3. Os.txt를 os\_copy.txt명으로 복사 : `cp os.txt os_cpoy.txt`
4. Os\_copy.txt의 권한을 모든 대상에게 읽기만 부여 : `chmod 444 os_copy.txt`

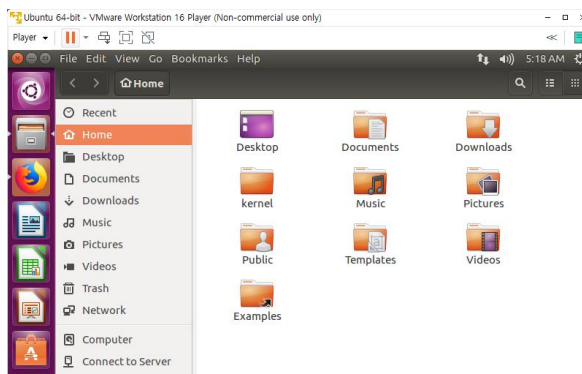
```
os2018202074@ubuntu:~/assignment1$ vi os.txt
os2018202074@ubuntu:~/assignment1$ cat os.txt
os_2018202074
os2018202074@ubuntu:~/assignment1$
```

```
os2018202074@ubuntu: ~/assignment1
os_2018202074
~
~
~
:wq
```

5. Os.txt에 os\_본인학번 작성 후 터미널에 출력

- vi os.txt : os.txt를 vi 편집기 환경으로 열기
- vi 편집기 내부에서 i를 입력 모드로 입장.
- Os\_2018202074를 입력모드에서 기술하고 :wq를 통해 저장 후 vi 종료
- Cat os.txt를 통해 txt파일 내 적은 os\_2018202074 출력

## Assignment1-2-Linux command



```
os2018202074@ubuntu: ~/kernel
os2018202074@ubuntu:~$ cd kernel
os2018202074@ubuntu:~/kernel$ sudo wget https://cdn.kernel.org/pub/linux/kernel/
v4.x/linux-4.19.67.tar.xz
[sudo] password for os2018202074:
--2022-09-07 05:30:45-- 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 35.4MB/s in 2.8s
2022-09-07 05:30:48 (35.4 MB/s) - 'linux-4.19.67.tar.xz' saved [103291756/103291
756]

os2018202074@ubuntu:~/kernel$
```

위 그림과 같이 home에 kernel file을 만들고 해당 환경에서 이를 진행하고자 하였다. Cd를 통해 이동후 sudo wget https:... 명령어를 이용해 해당 링크에서 다운로드 받을 수 있는 Kernel source를 다운받은 것을 확인할 수 있다.(wget : 웹상 파일 다운로드 명령어)

```
os2018202074@ubuntu: ~/kernel
os2018202074@ubuntu:~/kernel$ ls
linux-4.19.67.tar.xz
os2018202074@ubuntu:~/kernel$ tar -jxvf linux-4.19.67.tar.xz
```

이후 받은 파일을 확인하고 tar 명령어를 통해 압축을 해제하였다.

옵션은 j, x, v, f로 순서대로 (bzip2압축 적용 옵션, tar에서 파일 추출, 처리과정 나열, 대상 지정)을 의미한다.

```
os2018202074@ubuntu: ~/kernel
linux-4.19.67/virt/kvm/arm/vgic/vgic-kvm-device.c
linux-4.19.67/virt/kvm/arm/vgic/vgic-mmio-v2.c
linux-4.19.67/virt/kvm/arm/vgic/vgic-mmio-v3.c
linux-4.19.67/virt/kvm/arm/vgic/vgic-mmio.c
linux-4.19.67/virt/kvm/arm/vgic/vgic-mmio.h
linux-4.19.67/virt/kvm/arm/vgic/vgic-v2.c
linux-4.19.67/virt/kvm/arm/vgic/vgic-v3.c
linux-4.19.67/virt/kvm/arm/vgic/vgic-v4.c
linux-4.19.67/virt/kvm/arm/vgic/vgic.c
linux-4.19.67/virt/kvm/arm/vgic/vgic.h
linux-4.19.67/virt/kvm/asynic_pf.c
linux-4.19.67/virt/kvm/asynic_pf.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/
linux-4.19.67/virt/lib/Kconfig
linux-4.19.67/virt/lib/Makefile
linux-4.19.67/virt/lib/irqbypass.c
os2018202074@ubuntu: ~/kernel$
```

kernel source이 압축 해제되었다.

```
os2018202074@ubuntu: ~/kernel/linux-4.19.67
os2018202074@ubuntu:~/kernel$ ls
linux-4.19.67  linux-4.19.67.tar.xz
os2018202074@ubuntu:~/kernel$ cd linux-4.19.67/
os2018202074@ubuntu:~/kernel/linux-4.19.67$ ls
arch      CREDITS   firmware  ipc        lib        mm         scripts   usr
block     crypto    fs         Kbuild     LICENSES   net        security  virt
certs     Documenta include   Kconfig    MAINTAINERS  README    sound
COPYING   drivers   init       kernel     Makefile    samples   tools
os2018202074@ubuntu:~/kernel/linux-4.19.67$ vi Makefile
```

```
os2018202074@ubuntu: ~/kernel/linux-4.19.67
# SPDX-License-Identifier: GPL-2.0
VERSION = 4
PATCHLEVEL = 19
SUBLEVEL = 67
EXTRAVERSION =
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
"Makefile" 1736 lines, 60009 characters

os2018202074@ubuntu: ~/kernel/linux-4.19.67
# SPDX-License-Identifier: GPL-2.0
VERSION = 4
PATCHLEVEL = 19
SUBLEVEL = 67
EXTRAVERSION = -2018202074
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)

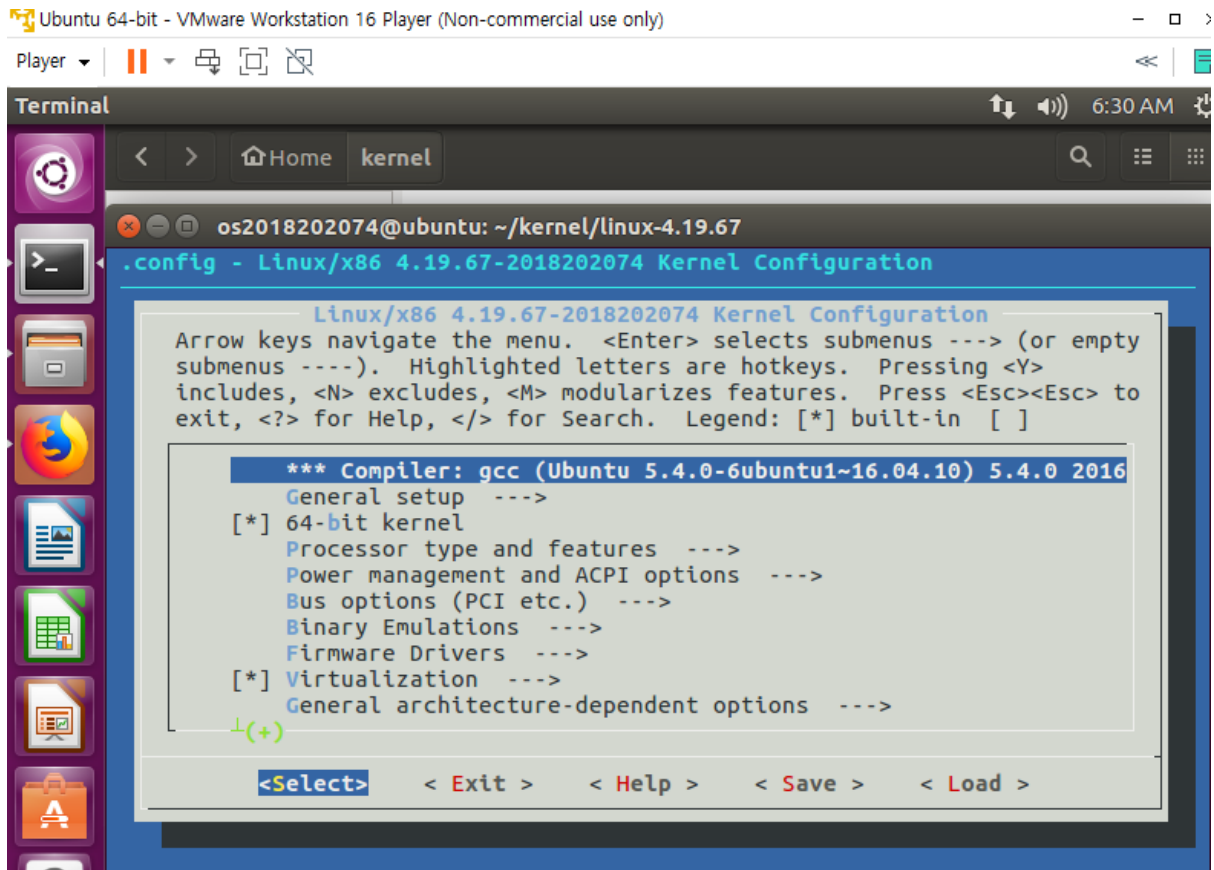
# Avoid funny character set dependencies
:wq
```

이후 ls를 통해 내부 파일을 확인, 압축이 풀린 것을 확인 후 cd를 통해 내부로 들어가 ls로 Makefile을 확인했다.vi Makefile를 통해 Makefile을 vi환경에서 열고 i를 통해 생성된 Make file내 부의 Extraversion에 -본인학번(-2018202074)를 추가하였다.

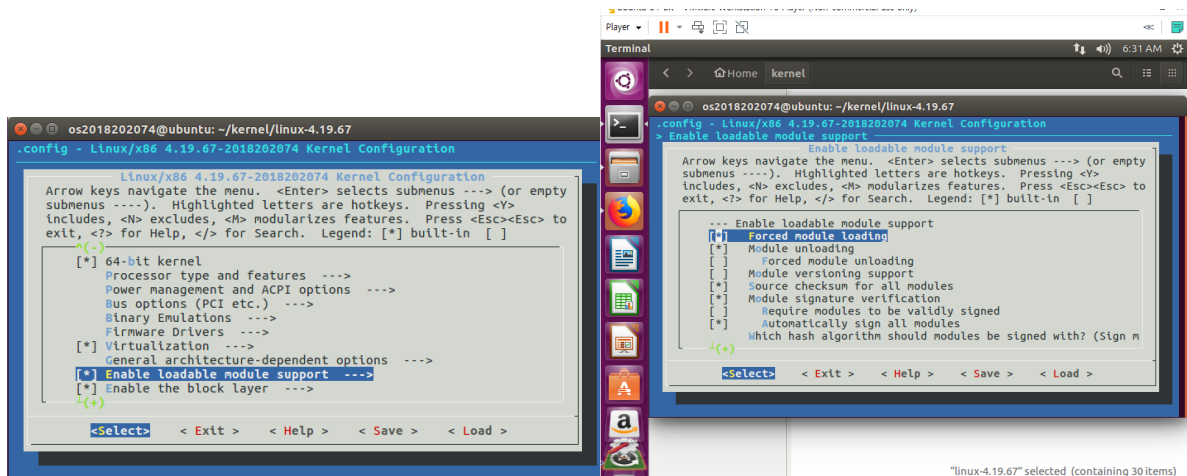
```
os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo apt install build-essential libncurses5-dev bison flex libssl-dev libelf-dev
[sudo] password for os2018202074:
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.1ubuntu2).
The following additional packages will be installed:
  libbison-dev libelf-dev libelf1-dev libncurses5-dev libncursesw5-dev libssl1.0.0 libzlib-dev
Suggested packages:
  bison-doc ncurses-doc
The following NEW packages will be installed:
  bison flex libbison-dev libelf-dev libelf1-dev libncurses5-dev libncursesw5-dev libssl-dev libzlib-dev
The following packages will be upgraded:
  libelf1 libssl1.0.0 zlib
3 upgraded, 12 newly installed, 0 to remove and 496 not upgraded.
Need to get 5,183 kB of archives.
After this operation, 15.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo make menuconfig
Preparing to unpack .../libssl-doc_1.0.2g-1ubuntu4.20_all.deb ...
Unpacking libssl-doc (1.0.2g-1ubuntu4.20) ...
Processing triggers for libc-bin (2.23-0ubuntu10) ...
Setting up libbison-dev:amd64 (2.7.5-1) ...
Setting up libzlib-dev:amd64 (1:1.2.8.dfsg-2ubuntu4.3) ...
Setting up libelf-dev:amd64 (0.165-3ubuntu1.2) ...
Setting up flex (2.6.0-11) ...
Setting up libbison1.0:amd64 (2.7.5-1) ...
Setting up libelf1:amd64 (0.165-3ubuntu1.2) ...
Setting up libbison-dev:amd64 (2.7.5-1) ...
Setting up bison (2.7.5-1) ...
update-alternatives: using /usr/bin/bison.yacc to provide /usr/bin/yacc (yacc) in auto mode
Setting up libelf-dev:amd64 (0.165-3ubuntu1.2) ...
Setting up libbison-dev:amd64 (2.7.5-1) ...
Setting up libzlib-dev:amd64 (1:1.2.8.dfsg-2ubuntu4.3) ...
Setting up libssl-dev:amd64 (1.0.2g-1ubuntu4.20) ...
Setting up libssl-doc (1.0.2g-1ubuntu4.20) ...
Processing triggers for libc-bin (2.23-0ubuntu10) ...
os2018202074@ubuntu:~/kernel/linux-4.19.67$
```

kernel compile을 위해 위에 해당하는 명령어를 통해 download를 진행하고 환경설정을 위해 sudo make menuconfig를 진행하였다. 이 명령어는 소스 코드 컴파일에 사용하는 Linux 소스를 제어할 수 있는 도구 중 하나인 menuconfig를 생성하여 Linux기능의 선택을 도와준다.

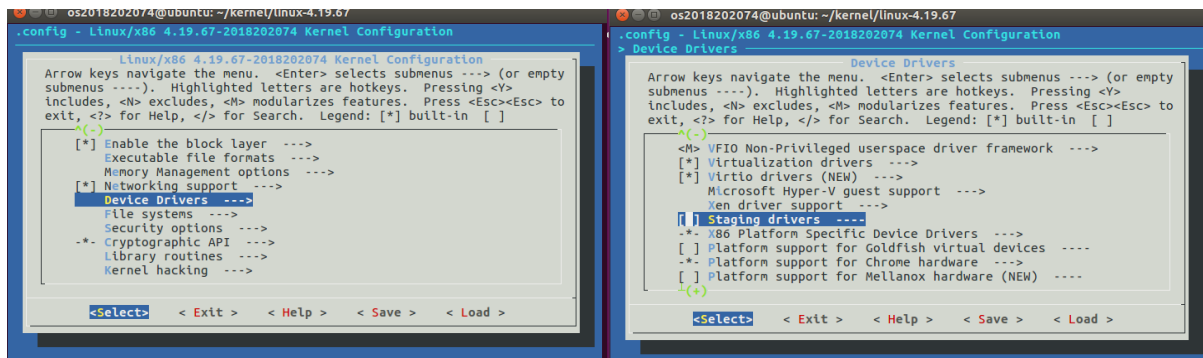


Menuconfig가 안정적으로 실행된 것을 확인하였다.

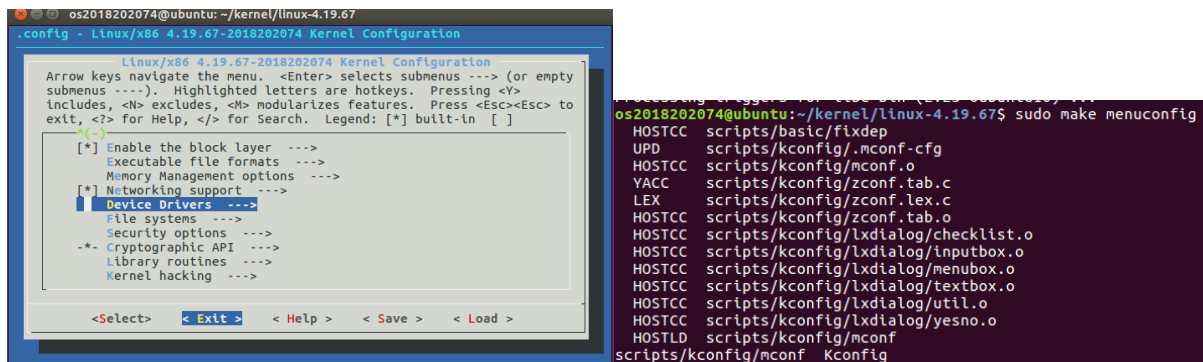
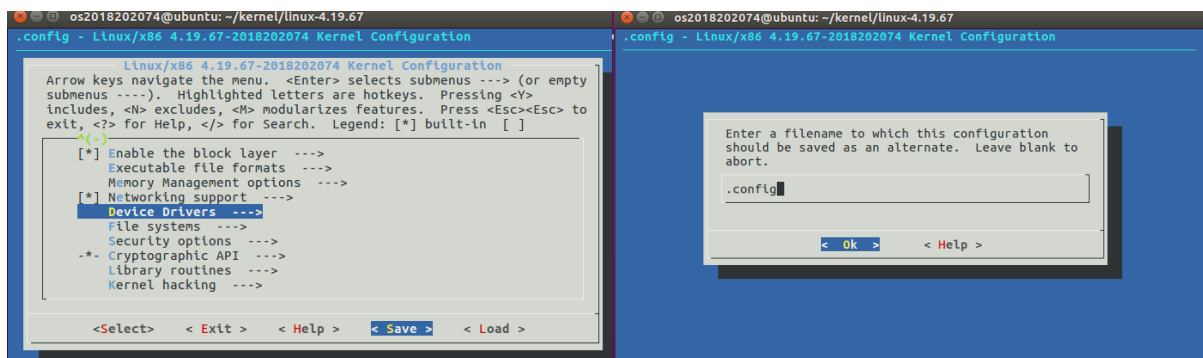


Enable loadable module support-> Forced module loading을 활성화 시켜 커널 모듈 적재 시 발생할 수 있는 문제를 해결한다.





마찬가지로 컴파일 시 문제가 될 수 있는 모듈을 제거하기 위해 Device Drivers로 들어가 Staging drivers를 비활성화 해준다.



```
*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

os2018202074@ubuntu:~/kernel/linux-4.19.67$
```

이후 밖으로 나와 Save를 진행하고 ok를 통해 저장 후 exit를 통해 menuconfig환경에서 나왔다.

```
os2018202074@ubuntu:~/kernel/linux-4.19.67$ make -j 8
Makefile:590: include/config/auto.conf: No such file or directory
Makefile:621: include/config/auto.conf.cmd: No such file or directory
HOSTCC scripts/kconfig/conf.o
HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --synconfig Kconfig
SYSTBL arch/x86/include/generated/asm/syscalls_32.h
SYSTBL arch/x86/include/generated/asm/syscalls_64.h
SYSHDR arch/x86/include/generated/asm/unistd_32_ia32.h
SYSHDR arch/x86/include/generated/asm/unistd_64_x32.h
HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_32.h
HOSTCC arch/x86/include/generated/uapi/asm/hwcap.h
```

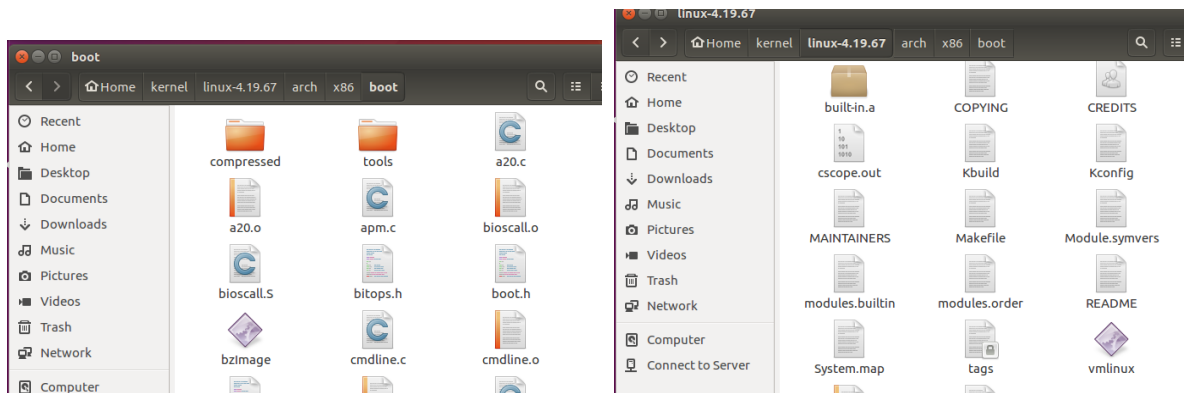


이후 make -j 8 (할당 Core수의 2배)를 통해 한번에 수행 가능한 명령의 수를 지정해주었다.

```
os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo make modules_install
INSTALL arch/x86/crypto/aes-x86_64.ko
INSTALL arch/x86/crypto/aesni-intel.ko
INSTALL arch/x86/crypto/blowfish-x86_64.ko
INSTALL arch/x86/crypto/camellia-aesni-avx-x86_64.ko
INSTALL arch/x86/crypto/camellia-aesni-avx2.ko
INSTALL arch/x86/crypto/camellia-x86_64.ko
INSTALL arch/x86/crypto/cast5-avx-x86_64.ko
INSTALL arch/x86/crypto/cast6-avx-x86_64.ko
INSTALL arch/x86/crypto/chacha20-x86_64.ko
INSTALL arch/x86/crypto/crc32-pclmul.ko
INSTALL arch/x86/crypto/crct10dif-pclmul.ko
INSTALL arch/x86/crypto/des3_ede-x86_64.ko
INSTALL arch/x86/crypto/ghash-clmulni-intel.ko
INSTALL arch/x86/crypto/glmul-helper.ko

os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo make install
sh ./arch/x86/boot/install.sh 4.19.67-2018202074 arch/x86/boot/bzImage \
    System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 4.19.67-2018202074
/boot/vmlinuz-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.19.67-2018202074 /
boot/vmlinuz-4.19.67-2018202074
update-initramfs: Generating /boot/initrd.img-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/pm-utils 4.19.67-2018202074 /boot/vm
linux-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 4.19.67-20182020
74 /boot/vmlinuz-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/update-notifier 4.19.67-2018202074 /
boot/vmlinuz-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 4.19.67-2018202074 /b
oot/vmlinuz-4.19.67-2018202074
Generating grub configuration file
```

이후 sudo make modules\_install을 이용해 컴파일 완료된 module을 설치하였다.



이후 Kernel Image와 System Map을 찾은 후, 한 파일로 복사하였다.

```
os2018202074@ubuntu:~/kernel/linux-4.19.67/boot$ ls
bzImage  System.map

os2018202074@ubuntu:~/kernel/linux-4.19.67/boot$ sudo cp bzImage /boot
os2018202074@ubuntu:~/kernel/linux-4.19.67/boot$ sudo cp System.map /boot
os2018202074@ubuntu:~/kernel/linux-4.19.67/boot$
```

이후 Cp를 이용해 /boot로 복사해주었다.

```
os2018202074@ubuntu:/boot$ ls
abi-4.15.0-29-generic          memtest86+_multiboot.bin
bzImage                       retpoline-4.15.0-29-generic
config-4.15.0-142-generic      System.map
config-4.15.0-29-generic       System.map-4.15.0-142-generic
config-4.19.67-2018202074     System.map-4.15.0-29-generic
config-4.19.67-2018202074.old System.map-4.19.67-2018202074
grub                          System.map-4.19.67-2018202074.old
initrd.img-4.15.0-142-generic  vmlinuz-4.15.0-142-generic
initrd.img-4.15.0-29-generic  vmlinuz-4.15.0-29-generic
initrd.img-4.19.67-2018202074 vmlinuz-4.19.67-2018202074
memtest86+.bin                vmlinuz-4.19.67-2018202074.old
memtest86+.elf
```

ls를 통해 무사히 복사된 것을 확인할 수 있었다. 이들은 Grub boot loader에 자동 등록된다.

```
os2018202074@ubuntu: ~/kernel/linux-4.19.67
os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo vi /etc/default/grub
```

```
os2018202074@ubuntu:~/kernel/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=true
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 locale=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
#"/etc/default/grub" 34 lines, 1300 characters

os2018202074@ubuntu:~/kernel/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 locale=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
:wq!
```

이후 grub 파일 수정을 위해 vi /etc/default/grub을 통해 grub을 vi환경에서 열고 GRUB\_HIDDEN\_TIMEOUT=0을 주석 처리하며 동시에 GRUB\_HIDDEN\_TIMEOUT을 false로 변경해주었다.

Reboot를 진행하고 Grub boot loader에서 ubuntu를 선택하였다. 이후 uname -r명령어를 통해 kernel release를 확인해 변경이 잘 된 것을 확인하였다.

```
os2018202074@ubuntu:~/kernel/linux-4.19.67$ uname -r
4.19.67-2018202074
os2018202074@ubuntu:~/kernel/linux-4.19.67$
```

### Assignment1-3-Linux kernel message 출력

```
os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo apt install exuberant-ctags
[sudo] password for os2018202074:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  vim | nvi | vile | emacs
The following NEW packages will be installed:
  exuberant-ctags
0 upgraded, 1 newly installed, 0 to remove and 496 not upgraded.
Need to get 126 kB of archives.
After this operation, 341 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/main amd64 exuberant-ctags amd64 1:5.9~svn20110310-11 [126 kB]
Fetched 126 kB in 1s (104 kB/s)
Selecting previously unselected package exuberant-ctags.
(Reading database ... 179328 files and directories currently installed.)
Preparing to unpack .../exuberant-ctags_1%3a5.9~svn20110310-11_amd64.deb ...
Unpacking exuberant-ctags (1:5.9~svn20110310-11) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up exuberant-ctags (1:5.9~svn20110310-11) ...
update-alternatives: using /usr/bin/ctags-exuberant to provide /usr/bin/ctags (c
tags) in auto mode
update-alternatives: using /usr/bin/ctags-exuberant to provide /usr/bin/etags (e
tags) in auto mode
os2018202074@ubuntu:~/kernel/linux-4.19.67$
```

Ctags를 사용하기 위해 sudo apt install exuberant-ctags를 통해 설치를 진행했다.

```
os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo ctags -R
os2018202074@ubuntu:~/kernel/linux-4.19.67$ ls
arch      Documentation  Kconfig      modules.order  System.map
block     drivers        kernel        Module.symvers  tags
boot      firmware      lib           net             tools
built-in.a fs             LICENSES     README          usr
certs     include        MAINTAINERS  samples         virt
COPYING   init           Makefile     scripts         vmlinux
CREDITS   ipc            mm           security        vmlinux-gdb.py
crypto    Kbuild        modules.builtin  sound          vmlinux.o
os2018202074@ubuntu:~/kernel/linux-4.19.67$
```

이후 ctags -R을 통해 DB를 생성하였다. 내부에 tags가 있는 걸 확인할 수 있다.

```
os2018202074@ubuntu: ~/kernel/linux-4.19.67
os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo vi -t task_struct
```

잘 작동되는 지 확인을 위해 vi환경에서 task\_struct 함수 위치에서 열리도록 해보았다.

```

os2018202074@ubuntu: ~/kernel/linux-4.19.67
};

enum perf_event_task_context {
    perf_invalid_context = -1,
    perf_hw_context = 0,
    perf_sw_context,
    perf_nr_task_contexts,
};

struct wake_q_node {
    struct wake_q_node *next;
};

struct task_struct {
#ifdef CONFIG_THREAD_INFO_IN_TASK
    /*
     * For reasons of header soup (see current_thread_info()), this
     * must be the first element of task_struct.
     */
    struct thread_info          thread_info;
#endif
    /* -1 unrunnable, 0 runnable, >0 stopped: */
    volatile long               state;

    /*
     * This begins the randomizable portion of task_struct. Only
     * scheduling-critical items should be added above here.
     */
}
#include/linux/sched.h" 1911 lines, 53889 characters

```

올바르게 작동하는 것을 확인할 수 있었다.

이후 sudo apt install cscope를 통해 cscope를 추가적으로 설치해주었다.

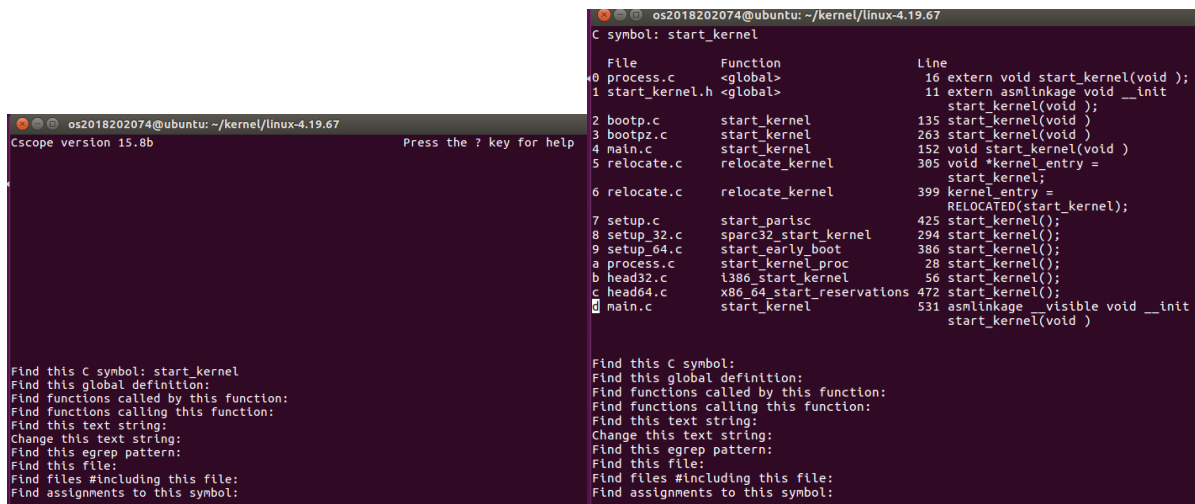
```

os2018202074@ubuntu: ~/kernel/linux-4.19.67
os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo apt install cscope
[sudo] password for os2018202074:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  cscope-el
The following NEW packages will be installed:
  cscope
0 upgraded, 1 newly installed, 0 to remove and 496 not upgraded.
Need to get 207 kB of archives.
After this operation, 1,239 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/universe amd64 cscope amd64 15.8b-1build1 [207 kB]
Fetched 207 kB in 1s (113 kB/s)
Selecting previously unselected package cscope.
(Reading database ... 179334 files and directories currently installed.)
Preparing to unpack .../cscope_15.8b-1build1_amd64.deb ...
Unpacking cscope (15.8b-1build1) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up cscope (15.8b-1build1) ...
os2018202074@ubuntu:~/kernel/linux-4.19.67$

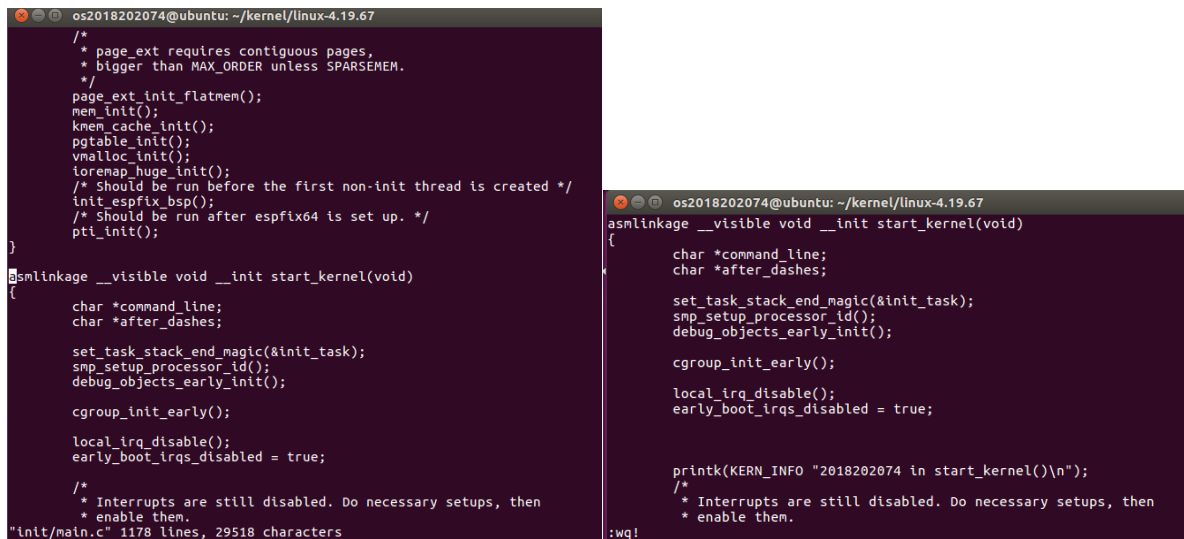
os2018202074@ubuntu:~/kernel/linux-4.19.67$ cscope -R

```

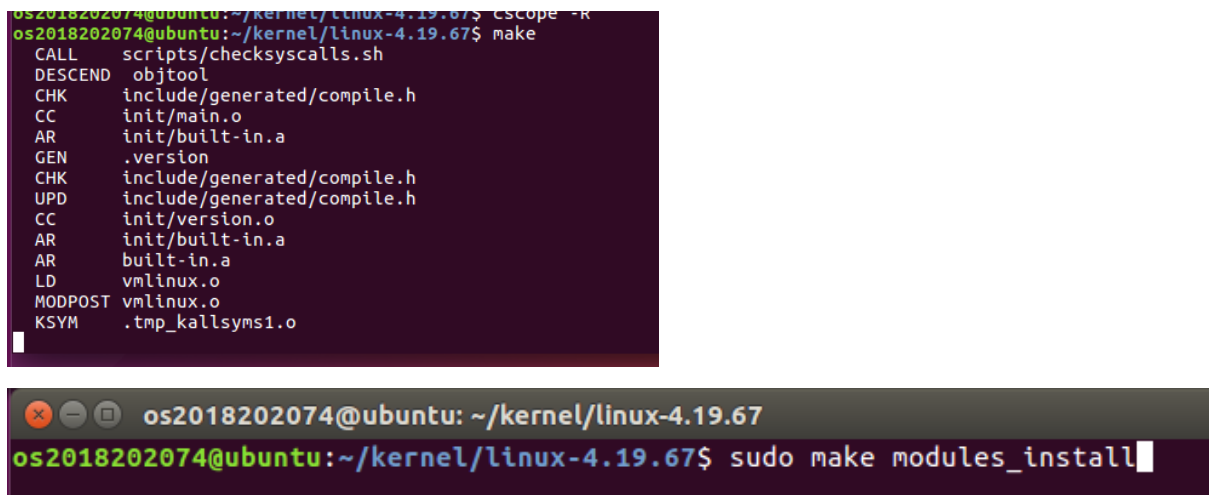
이후 cscope -R을 통해 DB를 구축하고 동시에 cscope를 실행해보았다.



정상적으로 내부로 들어가진 것을 확인하였다. 이후 find this Cy symbol에 start\_kernel을 입력하여 해당 main.c 를 발견하였다.



내부로 잘 들어온 것을 확인했다. 이후 vi환경을 바탕으로 내부의 printk를 추가하였다. 이는 이후 출력이 잘 동작하는지 확인하기 위함이다.





```

INSTALL sound/usb/line6/snd-usb-variax.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 4.19.67-2018202074
os2018202074@ubuntu:~/kernel/linux-4.19.67$

```

```

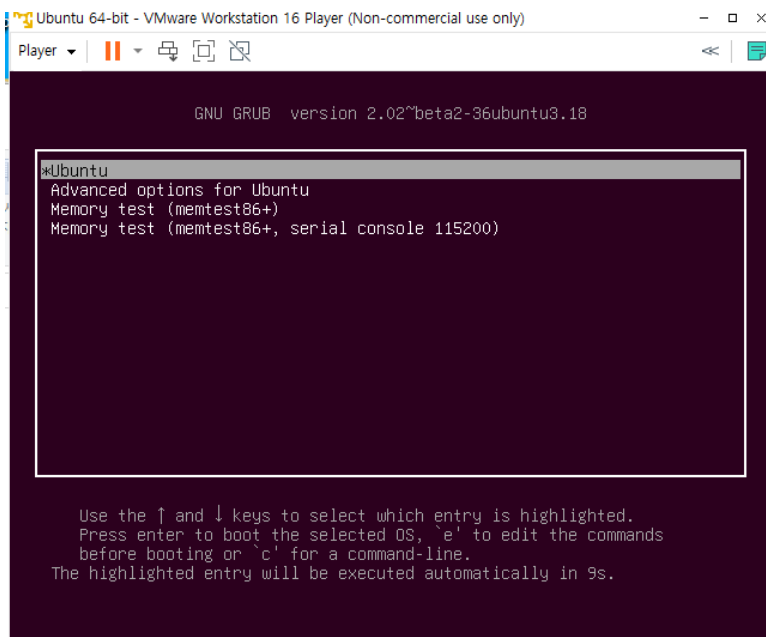
os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo make install
sh ./arch/x86/boot/install.sh 4.19.67-2018202074 arch/x86/boot/bzImage \
    System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 4.19.67-2018202074
/boot/vmlinuz-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.19.67-2018202074 /
boot/vmlinuz-4.19.67-2018202074
update-initramfs: Generating /boot/initrd.img-4.19.67-2018202074

```

```

os2018202074@ubuntu: ~/kernel/linux-4.19.67
boot/vmlinuz-4.19.67-2018202074
update-initramfs: Generating /boot/initrd.img-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/pm-utils 4.19.67-2018202074 /boot/vm
linux-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 4.19.67-20182020
74 /boot/vmlinuz-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/update-notifier 4.19.67-2018202074 /
boot/vmlinuz-4.19.67-2018202074
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 4.19.67-2018202074 /b
oot/vmlinuz-4.19.67-2018202074
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-4.19.67-2018202074
Found initrd image: /boot/initrd.img-4.19.67-2018202074
Found linux image: /boot/vmlinuz-4.19.67-2018202074.old
Found initrd image: /boot/initrd.img-4.19.67-2018202074
Found linux image: /boot/vmlinuz-4.15.0-29-generic
Found initrd image: /boot/initrd.img-4.15.0-29-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
os2018202074@ubuntu:~/kernel/linux-4.19.67$ reboot

```



이후 make, make modules\_install, make install을 통해 module compile을 진행하여 해당 변경을 지정해주고 reboot를 진행하였다.



```
os2018202074@ubuntu: ~  
os2018202074@ubuntu:~$ dmesg | grep "2018202074 in start_kernel()"  
[ 0.000000] 2018202074 in start_kernel()  
os2018202074@ubuntu:~$
```

해당 문자열을 찾는 dmesg | grep을 이용해 다음과 같이 적용되었음을 확인하였다.

```
os2018202074@ubuntu: ~  
os2018202074@ubuntu:~$ cscope -R
```

과제를 진행하기 위해 다시 cscope를 진행시켰다.

```
os2018202074@ubuntu: ~/kernel/linux-4.19.67  
File Edit View Search Terminal Help  
Text string: Linux agp  


| File      | Line |
|-----------|------|
| backend.c | 338  |

  
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:
```

Linux agp라는 문자열을 포함하는 것을 위치를 검색하니 다음과 같은 위치가 출력되었다.

```

os2018202074@ubuntu: ~/kernel/linux-4.19.67
File Edit View Search Terminal Help

    mem_init();
    kmem_cache_init();
    pgtable_init();
    vmalloc_init();
    ioremap_huge_init();
    /* Should be run before the first non-init thread is created */
    init_espfix_bsp();
    /* Should be run after espfix64 is set up. */
    pti_init();
}

asmlinkage __visible void __init start_kernel(void)
{
    char *command_line;
    char *after_dashes;

    set_task_stack_end_magic(&init_task);
    smp_setup_processor_id();
    debug_objects_early_init();

    cgroup_init_early();

    local_irq_disable();
:ts agp

```

```

# pri kind tag file
1 F m agp drivers/gpu/drm/nouveau/include/nvkm/subdev/pci.h
    struct:nvkm_pci typeref:struct:nvkm_pci::__anon7038
    } agp;
2 F m agp drivers/gpu/drm/nouveau/nouveau_drv.h
    struct:nouveau_drm typeref:struct:nouveau_drm::__anon7022
    } agp;
3 F m agp drivers/gpu/drm/via/via_verifier.h
    struct::__anon7734
    int agp;
4 F m agp include/drm/drm_device.h
    struct:drm_device typeref:struct:drm_device::drm_agp_head
    struct drm_agp_head *agp; /**< AGP data */
5 FS m agp drivers/char/agp/backend.c
    struct::__anon4932
    static const struct { int mem, agp; } maxes_table[] = {
Type number and <Enter> (empty cancels): 5

```

Ctags의 경우 :ts agp를 이용, agp가 포함된 위치들을 확인했다. 5번째 검색결과에서 cscope에서의 결과와 동일한 파일인 backend.c가 나온 것을 확인해 5를 입력해 내부로 들어가보았다.

```

os2018202074@ubuntu: ~
if (list_empty(&agp_bridges))
    agp_frontend_cleanup();
module_put(bridge->driver->owner);
}
EXPORT_SYMBOL_GPL(agp_remove_bridge);

int agp_off;
int agp_try_unsupported_boot;
EXPORT_SYMBOL(agp_off);
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)
{
}

"kernel/linux-4.19.67/drivers/char/agp/backend.c" 366 lines, 9082 characters

```

```

os2018202074@ubuntu: ~/kernel/linux-4.19.67
if (list_empty(&agp_bridges))
    agp_frontend_cleanup();
module_put(bridge->driver->owner);
}
EXPORT_SYMBOL_GPL(agp_remove_bridge);

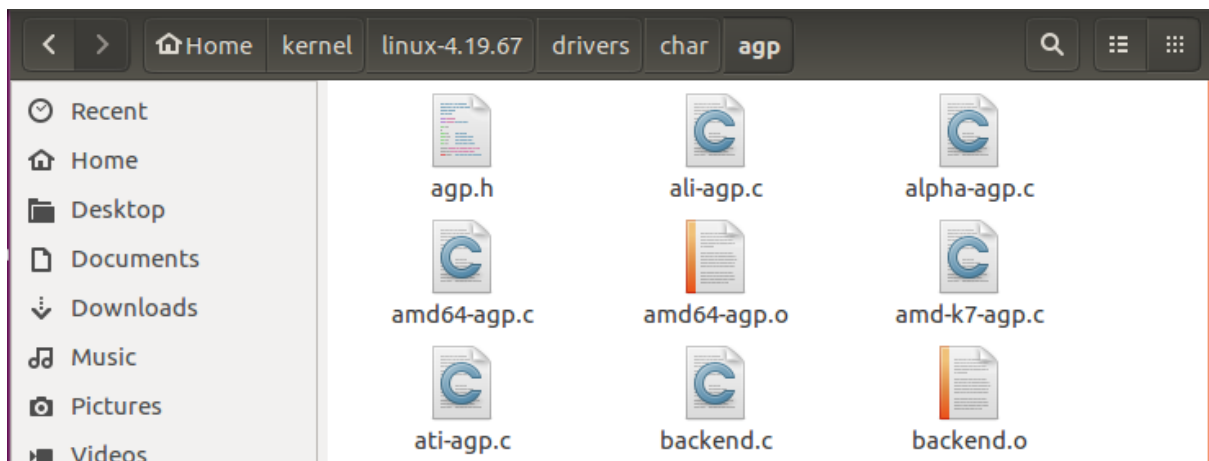
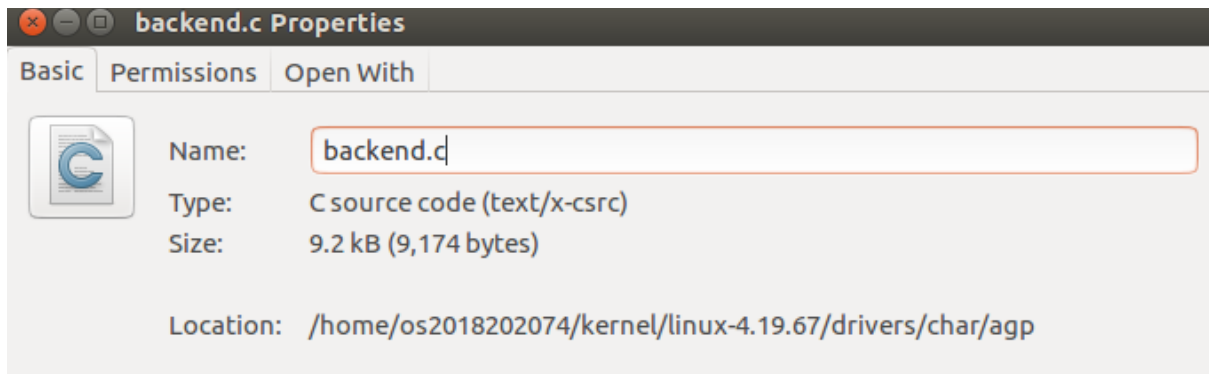
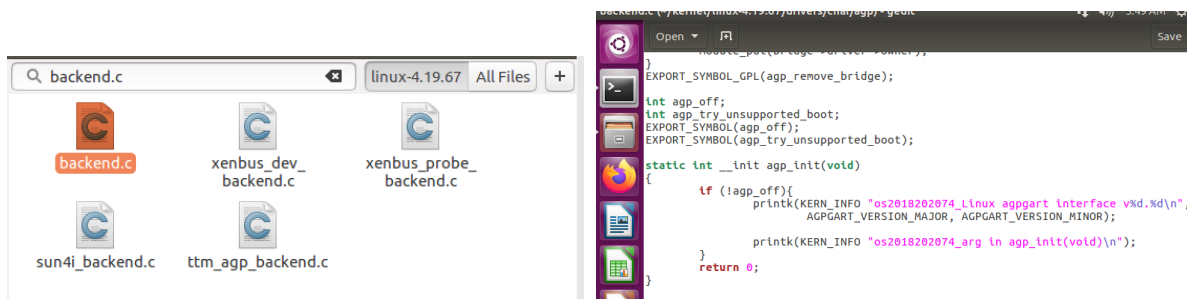
int agp_off;
int agp_try_unsupported_boot;
EXPORT_SYMBOL(agp_off);
EXPORT_SYMBOL(agp_try_unsupported_boot);

static int __init agp_init(void)
{
    if (!agp_off){
        printk(KERN_INFO "os2018202074_Linux agpgart interface v%d.%d\n",
                AGPGART_VERSION_MAJOR, AGPGART_VERSION_MINOR);
        printk(KERN_INFO "os2018202074_arg in agp_init(void)\n");
    }
    return 0;
}

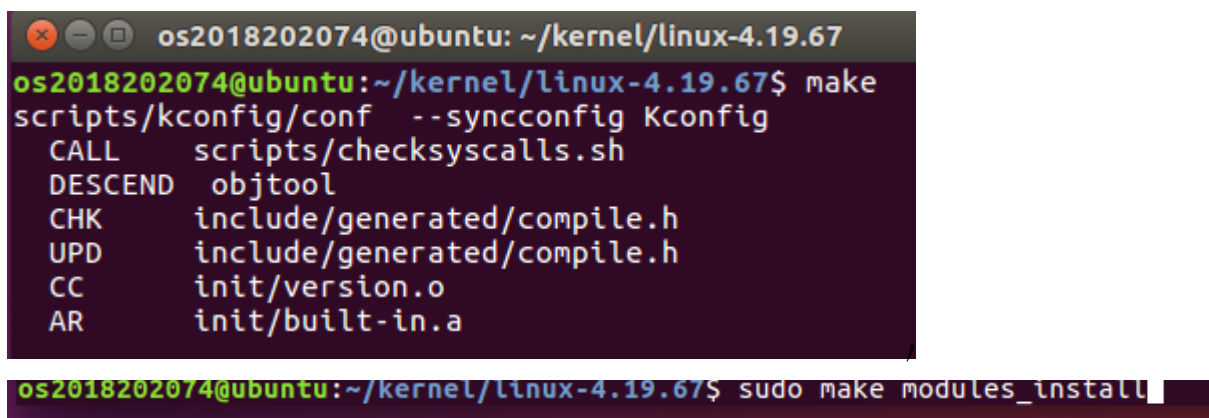
:wq!

```

Agp\_init의 내부를 수정한다. 이때 printk를 사용하여 기존에 있던 출력 앞에 os학번을 추가하고 조건에 맞게 문자열 하나가 더 출력되도록 하였다. (os본인학번\_arg in agp\_init(void))



해당 파일이 수정된 것을 확인할 수 있었고, 소스코드의 path 또한 찾을 수 있었다.



```
INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL sound/x86/snd-hdmi-lpe-audio.ko
INSTALL virt/lib/irqbypass.ko
DEPMOD 4.19.67-2018202074
os2018202074@ubuntu:~/kernel/linux-4.19.67$ sudo make install
```

```
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
os2018202074@ubuntu:~/kernel/linux-4.19.67$ reboot
```

이전과 동일하게 make, make modules\_install, make install을 통해 module compile을 진행하여 해당 변경을 지정해주고 reboot를 진행하였다.

```
os2018202074@ubuntu: ~
File Edit View Search Terminal Help
os2018202074@ubuntu:~$ dmesg | grep "os2018202074" -n
2:[    0.000000] Linux version 4.19.67-2018202074 (os2018202074@ubuntu) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.12)) #4 SMP Mon Sep 12 05:58:22 PDT 2022
1411:[   21.132934] os2018202074_Linux agpgart interface v0.103
1412:[   21.132938] os2018202074_arg in agp_init(void)
os2018202074@ubuntu:~$
```

검색의 결과로 다음과 같이 출력되는 것을 확인할 수 있다.

## 고찰

이번 과제를 통해 VMware와 Ubuntu의 재설치를 진행하며 온라인 환경에서 가상환경을 구축하는 법을 다시금 실습해 볼 수 있었습니다. 또한 기본 linux command에 대한 문제를 풀며 시스템 프로그래밍 과목을 복습하는 좋은 기회가 되었습니다. Assignment 1-2, 1-3을 진행하면서 kernel이 돌아갈 수 있는 환경을 구성하고 이를 통해 kernel release를 확인하는 `uname -r` 명령어에 학번을 띄우거나 `agp_init`의 출력을 변경하는 등의 행위를 통해 kernel의 대략적인 구조 및 이를 보조하는 `cscope`, `ctags`를 이용하는 방법을 알 수 있었습니다. 이를 바탕으로 남은 2학기 동안 kernel을 다뤄보며 kernel에 대해 배워가고자 합니다.

## Reference

강의자료 참고

2022-2\_OSLab\_Assignment\_1.pdf

2022-2\_OSLab\_03\_Linux\_Kernel.pdf